Entwicklung eines webbasierten Werkzeuges zur Technologieplanung für die Integration in einen ganzheitlichen Planungsprozess

Von der Fakultät für Elektro- und Informationstechnik der
Hochschule Mittweida (FH)

genehmigtes
Bakkalaureat

zur Erlangung des akademischen Grades

Bachelor of Engineering
(B. Eng.)

vorgelegt

von	Frank Schaarschmidt	
geboren am	31. Oktober 1984	in Karl – Marx – Stadt
eingereicht am	17. Juli 2011	

Gutachter:	Prof. F. Zimmer
	Dr.-Ing. U. Günther

Mittweida, den 14. Juli 2011

Bibliographische Angaben und Referat

Schaarschmidt Frank

Thema

„Entwicklung eines webbasierten Werkzeuges zur Technologieplanung für die Integration in einen ganzheitlichen Planungsprozess“

Bachelorarbeit an der Fakultät für Elektro- und Informationstechnik der Hochschule Mittweida, Chemnitz, 14. Juli 2011

60 Seiten, 27 Abbildungen, 6 Tabelle, 23 Literaturzitate, 2 Anlagen

Referat

Im Rahmen dieser Bachelorarbeit wird ein webbasiertes Werkzeug zur Technologieplanung entwickelt. Dazu werden zunächst in Form eines Pflichtenheftes Anforderungen an das System definiert. Es folgt die theoretische Betrachtung spezifischer Merkmale von Web-Anwendungen und die Abgrenzung zur konventionellen Software-Entwicklung. Basierend auf den gewonnenen Erkenntnissen wird ein geeignetes Vorgehensmodell ausgewählt. Daraufhin erfolgt die schrittweise Konzeption der Anwendung hinsichtlich Architektur, Benutzeroberfläche und Datenverwaltung. Nach Vorstellung der eingesetzten technischen Hilfsmittel wird das Konzept mit Hilfe einer prototypingorientierten Entwicklungsmethodik, realisiert. Abschließend erfolgt ein kurzer Ausblick auf Verbesserungs- und Erweiterungsmöglichkeiten der Web-Anwendung.

Erklärung zur selbstständigen Anfertigung der Arbeit

Ich versichere, dass ich die Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Wörtlich oder sinngemäß aus anderen Quellen übernommene Textstellen, Bilder, Tabellen u. a. sind unter Angabe der Herkunft kenntlich gemacht. Weiterhin versichere ich, dass diese Arbeit noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Chemnitz, am 14.07.2011

Frank Schaarschmidt

<<Aufgabenstellung einfügen>>

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS I

ABBILDUNGSVERZEICHNIS III

TABELLENVERZEICHNIS IV

ANLAGENVERZEICHNIS V

ABKÜRZUNGS- UND SYMBOLVERZEICHNIS VI

1 EINLEITUNG 1

1.1 MOTIVATION DER ARBEIT ... 1

1.2 ZIELSETZUNG UND VORGEHENSWEISE ... 2

2 ANFORDERUNGSANALYSE UND PROBLEMSTELLUNG 3

2.1 AUSGANGSSITUATION ... 4

2.2 PFLICHTENHEFT ... 6

2.3 WEB-ANWENDUNG ... 9

2.4 ANFORDERUNGEN AN EINEN ENTWICKLUNGSPROZESS FÜR WEB-ANWENDUNGEN ... 10

2.5 ENTWICKLUNGSBEZOGENE CHARAKTERISTIKA VON WEB-ANWENDUNGEN ... 12

2.5.1 PROJEKTMITARBEITER ... 12

2.5.2 TECHNISCHE INFRASTRUKTUR ... 13

2.5.3 ENTWICKLUNGSPROZESS ... 14

2.5.4 INTEGRATION ... 15

3 KONZEPTION EINER TECHNOLOGIEPLANUNGSANWENDUNG FÜR DAS PROJEKT SAMARA 16

3.1 VORGEHENSMODELLE DER SOFTWAREENTWICKLUNG UND DEREN ABLEITUNG FÜR WEB-ANWENDUNGEN ... 16

3.1.1 KLASSISCHES SEQUENTIELLES SOFTWARE-LIFE-CYCLE MODELL ... 18

3.1.2 PROTOTYPINGORIENTIERTES LIFE-CYCLE-MODELL ... 26

3.1.3 SPIRALMODELL ... 29

3.1.4 AUSWAHL EINES GEEIGNETEN VORGEHENSMODELLS FÜR DEN PROJEKTABLAUF ... 31

3.2 ARCHITEKTUR ... 33

3.2.1 GRUNDLAGEN ARCHITEKTUR ... 33

3.2.2 2-SCHICHTEN-ARCHITEKTUR DES TECHNOLOGIEPLANUNGSWERKZEUGS ... 36

3.3 GRAFISCHE BENUTZEROBERFLÄCHE (GUI) ... 37

3.4 DATENVERWALTUNG ... 42

4 TECHNISCHE GRUNDLAGEN 47

4.1 HYPERTEXT PREPROCESSOR (PHP) UND EINGESETZTE TECHNOLOGIEN ... 47

4.2 ENTWICKLUNGSUMGEBUNG ... 48

5 UMSETZUNG 50

5.1 NAVIGATION 50

5.2 DESIGN UND ERGONOMIE 51

5.3 VERARBEITUNG UND DARSTELLUNG DER PLANUNGSDATEN 55

6 ZUSAMMENFASSUNG UND AUSBLICK 57

6.1 ERGEBNISSE UND ABWEICHUNGEN 57

6.2 ERWEITERUNGS- UND VERBESSERUNGSVORSCHLÄGE 58

LITERATURVERZEICHNIS 59

GLOSSAR 60

ANLAGEN A1

A1 PFLICHTENHEFT A1

A2 DATENBANKSTRUKTUR A8

ABBILDUNGSVERZEICHNIS

Abbildung 1: Aufbau Pflichtenheft nach IEEE 830-1998 (vgl. [13], S. 169) 7
Abbildung 2: Software-Life-Cycle-Modell (nach [12], S. 18) 19
Abbildung 3: Schrittweise Konkretisierung nach dem „Topdown" – Prinzip 23
Abbildung 4: Prototyping orientiertes Life-Cycle-Modell nach POMBERGER (vgl. [12], S. 25) 27
Abbildung 5: Ablauf bei der Prototyperstellung 28
Abbildung 6: Spiralmodell nach BOEHM 29
Abbildung 7: Spiralförmiges prototypisches Vorgehensmodell 31
Abbildung 8: Einflussfaktoren auf die Entwicklung einer Architektur (nach [17]) 34
Abbildung 9: 2-Schichten-Architektur für Web-Anwendungen (nach[18]) 36
Abbildung 10: Strukturplan des Technologieplanungswerkzeugs 39
Abbildung 11: Anordnung der Funktionsbereiche 41
Abbildung 12: Aufbau des Datenbanksystem (vgl. [5], S. 720) 43
Abbildung 13: Der Entitätstyp Lieferanten in tabellarischer Form 44
Abbildung 14: Ausschnitt des Logischen Schemas der Technologieplanungsanwendung 44
Abbildung 15: Erzeugung einer Lieferanten-Tabelle mittels SQL 45
Abbildung 16: Navigations-Menü der Technologieplanungsanwendung(verkürzt dargestellt) 50
Abbildung 17: Ausschnitt aus dem Aufbau des Navigationsmenüs 50
Abbildung 18: Ausschnitt aus main.php: Einbinden der Web-Seiten über die content-Variable 51
Abbildung 19: Web-Design des Technologieplanungswerkzeugs 51
Abbildung 20: Umformungsmechanismus einer rationalen Zahl 53
Abbildung 21: Ausgabe und Präsentation von Fehlermeldungen 53
Abbildung 22: Beispiel Tooltip 53
Abbildung 23: Headerbereich der Technologieplanungsanwendung 54
Abbildung 24: Navigations- und Informationsdarstellungsbereich der Technologieplanungsanwendung 54
Abbildung 25: Zentrale Konfigurationsdatei für die DB-Verbindung 55
Abbildung 26: DB-Abfrage innerhalb einer PHP-Seite 55
Abbildung 27: Ausgabe der im Array gespeicherten Daten 56

Abbildung A1: Datenbankstruktur der Technologieplanungsanwendung A8

TABELLENVERZEICHNIS

Tabelle 1: Erfolgsstatistik von Softwareprojekten in den USA (vgl. [1], S. 236) 11

Tabelle A1: Übersicht Qualitätskriterien A4
Tabelle A2: Übersicht Produktfunktionen 1 ..... A5
Tabelle A3: Übersicht Produktfunktionen 2 ..... A5
Tabelle A4: Übersicht Produktfunktionen 3 ..... A6
Tabelle A5: Übersicht Produktfunktionen 4 ..... A6

ANLAGENVERZEICHNIS

A1 PFLICHTENHEFT A1

A2 DATENBANKSTRUKTUR A8

ABKÜRZUNGS- UND SYMBOLVERZEICHNIS

ANSI	American National Standards Institute
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheet
CVS	Concurrent Version System
DB	Datenbank
DBG	PHP Debugger
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DD	Data Dictionary
DDL	Data Definition Language
d.h.	das heißt
GmbH	Gesellschaft mit beschränkter Haftung
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
IT	Information Technology
J2EE	Java 2 Enterprise Edition
PC	Personal Computer
PDT	PHP Development Tools
PHP	Hypertext Preprocessor
RE	Requirements Engineering
sog.	so genannt
SQL	Structured Query Language
u.a.	unter anderem
usw.	und so weiter
vgl.	vergleiche
W3C	World Wide Web Consortium
XP	Extreme Programming
z.B.	zum Beispiel

1 EINLEITUNG

1.1 Motivation der Arbeit

Die zunehmende globale Verdichtung und die damit verbundenen Veränderungen in Wirtschaft und Gesellschaft haben vor allem für das produzierende Gewerbe einen härteren Wettbewerb und eine höhere Dynamik zur Folge. Wer auf den turbulenten Märkten heutzutage erfolgreich sein will, für den sind Reaktionsschnelligkeit, Flexibilität und vorausschauendes Handeln allein die nötigen Grundvoraussetzungen. Modernisierungs- und Umstrukturierungsmaßnahmen oder die Verlagerung ganzer Produktionsstraßen und sogar Unternehmenszweige sind keine Seltenheit wenn es darum geht sich auf kommende Wirtschaftsszenarien vorzubereiten.

Für die Produktionsplanung bzw. Technologieplanung stellen solche Maßnahmen immer neue Herausforderungen und Probleme dar, denn mit der Durchführung solcher Schritte verlieren oftmals ganze Produktionsprozesse ihre Gültigkeit. Mit der Neuanschaffung von Ressourcen, veränderter Umgebungs- und Zulieferbedingungen müssen Produktionsprozesse erneut projektiert und Technologien neu ausgewählt werden. Um weiter profitabel zu produzieren und konkurrenzfähig zu bleiben ist eines der wichtigsten Kriterien der Zeitfaktor. Qualitativ hochwertige Planungsergebnisse sollen trotz immer umfangreicherer und komplizierterer Produkte möglichst schnell vorliegen um die Produktion ohne große Verzögerung wieder aufnehmen zu können.

Technologieplaner haben heutzutage einen reichhaltigen Fundus an Software auf den sie zurückgreifen können und der sie bei ihren Planungsaufgaben unterstützt. Diese Software beschränkt sich allerdings hauptsächlich auf den Bereich der Tabellenkalkulation und es existiert im Moment noch keine Spezialsoftware für solche Anwendungsfälle. Daraus ergibt sich zusätzlicher Handlungsbedarf, denn immer wieder entstehen durch den sehr arbeitsteiligen Planungsprozess Probleme durch Inkonsistenz, Redundanz und mehrmaliges Erfassen von Planungsdaten. Vor allem beim Ändern von bestehenden Datensätzen steigt der entstehende Aufwand sehr schnell exponentiell an.

1.2 Zielsetzung und Vorgehensweise

Inhalt der Arbeit soll es sein, den Ansatz zu untersuchen ein webbasiertes Planungswerkzeug zu schaffen. Dieses soll die Aufgabe der Datenerfassung, Verarbeitung und Datenausgabe übernehmen. Zunächst soll dazu eine Anforderungsanalyse durchgeführt werden. Basierend auf den ermittelten Anforderungen an die Web-Anwendung, soll ein Entwicklungsprozess erstellt und ein entsprechendes Vorgehensmodell gewählt werden. Die gewonnenen Erkenntnisse sollen helfen ein Werkzeug zu entwickeln, dass den Umgang mit den Planungsdaten stark vereinfacht und Fehler minimiert.

Dabei soll zunächst ein entsprechendes Datenbankmodell ausgewählt werden, welches dann strukturiert bzw. entsprechend funktionaler Abhängigkeiten normalisiert wird. Basierend auf dieser Struktur soll eine Benutzeroberfläche in PHP bzw. HTML erstellt werden, die es ermöglicht über Formulare die Daten aufzunehmen und in der Datenbank zu speichern. Die Ausgabe und Weiterverarbeitung der Daten soll ebenfalls über die Web-Oberfläche erfolgen, dabei soll es möglich sein verschiedene Dokumente anzuzeigen und Berechnungen durchführen zu lassen. Außerdem soll für den Druck der entsprechenden Prozesspläne, Arbeitspläne, Maschineneinstellpläne etc. ein Druck – CSS erstellt werden, welches die Dokumente so formatiert, dass deren anschließende Verwendung direkt möglich ist. Abschließend soll das Werkzeug in einem Praxisprojekt erprobt werden. Dabei soll es um die Technologieplanung eines Wälzlagerwerkes in Samara (Russland) gehen, wo auf Grund von Modernisierungsmaßnahmen eine Neuplanung erforderlich ist und neue Prozesspläne sowie Produktionsdokumente benötigt werden.

Ziel der Bachelorarbeit ist die Bewertung möglicher Strategien sowie die Konzeption und Entwicklung eines Planungswerkzeuges zur Integration in eine heterogene Systemlandschaft. Mit Hilfe der Erkenntnisse aus der Vorbetrachtung soll am Beispiel eines Praxisprojektes eine webbasierte Plattform geschaffen werden, die es ermöglicht, Technologiedaten durchgängig und zum Teil projektübergreifend, im gesamten Planungsprozess zu nutzen. Dadurch sollen Technologieplanungsprojekte zukünftig einfacher realisiert und die Qualität der Planungsergebnisse gesteigert werden.

2 ANFORDERUNGSANALYSE UND PROBLEMSTELLUNG

Die Analyse der Anforderungen ist für den späteren Projektablauf von elementarer Bedeutung denn sie bildet die Grundlage für die spätere Vorgehensweise und die damit verbundenen, eingesetzten Methoden und Werkzeuge. GRÜNBACHER beschreibt die Anforderung als „… eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Systems“ ([1], S. 32). Nach dem IEEE 610.12-Standard ist sie folgendermaßen definiert:

1.) Eine Bedingung oder Fähigkeit, die von einer Person zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2.) Eine Bedingung oder Fähigkeit, die eine Software erfüllen oder besitzen muss, um einen Vertrag, eine Norm oder ein anderes, formell bestimmtes Dokument zu erfüllen.
3.) Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit wie in Punkt 1 oder 2. ([1], S. 32)

Man unterscheidet dabei zwischen *funktionalen* und *nicht funktionalen* Anforderungen. *Funktionale Anforderungen* beschreiben dabei die Dienste eines Systems, wie das System auf bestimmte Eingaben reagieren soll oder wie sich das System in verschiedenen Situationen verhalten soll.

Nichtfunktionale Anforderungen dagegen beschreiben Qualitätsmerkmale sowie Termin- und Kostenziele. Diese Anforderungen stellen quasi Beschränkungen des Systems dar, darunter fallen z.B. einzuhaltende Standards oder Beschränkungen des Entwicklungsprozesses. Nichtfunktionale Anforderungen beziehen sich nicht auf einzelne Funktionen oder Dienste, sondern meist auf das ganze System. ([1], S. 32; siehe auch [13], S. 152 ff.)

Das Ziel der Anforderungsanalyse ist das Erstellen und Warten eines Anforderungsdokuments, besser bekannt unter der Bezeichnung *Pflichtenheft.* Damit die Anforderungen in Form eines Pflichtenheftes spezifiziert werden können muss man sich zunächst mit der Ausgangssituation und projektspezifischen Gegebenheiten vertraut machen.

2.1 Ausgangssituation

In das Aufgabengebiet der HÖRMANN-RAWEMA GmbH in Chemnitz fallen als erfahrener und leistungsfähiger Engineering-Dienstleister auch Technologieplanungs- bzw. –umplanungsprojekte aus sämtlichen Industriezweigen. Solch ein Projekt bietet auch die Grundlage für die Entwicklung, des im Rahmen dieser Arbeit beschriebenen Planungswerkzeugs.

Im September 2010 bekam die Firma HÖRMANN-RAWEMA den Auftrag für ein Wälzlagerwerk der SPZ-Gruppe am Standort Samara in Russland eine umfassende Modernisierung (Re-Engineering) durchzuführen. Auf Grund der Tatsache, dass die am Standort Samara verfügbaren Produktionsanlagen überaltert waren, ergaben sich Mängel in der Einhaltung von Qualitätsstandards der erzeugten Produkte. Die vorhandenen Produktionsanlagen sollten deshalb durch neue Maschinen ersetzt werden. Dadurch verloren allerdings sämtliche Produktionsprozesse und -dokumente wie Arbeitspläne oder Einstellblätter ihre Gültigkeit, da diese an die neuen Maschinen angepasst bzw. neu erstellt werden mussten.

Das enorme Produktspektrum von ca. 900 verschieden Wälzlagern verschärft das Problem und macht den Aufwand, der zur Erstellung der Dokumente nötig ist, deutlich.
Zwar steht Technologieplanern heutzutage ein reichhaltiger Fundus an Software zur Verfügung, der sie bei Planungsaufgaben unterstützt, allerdings beschränkt sich dieser hauptsächlich auf den Bereich der Tabellenkalkulation oder zu umfangreiche Softwarepakete deren Anschaffungs-/Nutzenverhältnis sich nicht rentabilisiert. Die Probleme die dieser arbeitsteilige Planungsschritt mit sich bringt sind unübersehbar, das mehrmalige Erfassen von Datensätzen, die Inkonsistenz oder Redundanz von Planungsdaten stellen nur einige dar. Erfahrungen haben gezeigt, dass während des Planungsprozesses diese Dokumente erheblichen Änderungen unterworfen sind, deren Pflege und Wartung durch die enorme Menge an Dokumenten sich äußerst schwierig gestaltet und durch die unterschiedlichen Aktualisierungsstände der Daten Informationskonflikte auftreten. Zudem haben nur wenige Mitarbeiter den gesamten Überblick über die vorliegenden Daten und können aussagekräftige Äußerungen zum aktuellen Stand machen. So entstand der Wunsch nach einem System, dass den Vorgang der Datenwartung und Dokumenterstellung vereinfacht und teilautomatisiert.

Um die zuvor genannten Probleme zu lösen und die Qualität der Planungsergebnisse zu steigern, wurde entschieden im Rahmen einer Bachelorarbeit ein Planungswerkzeug mit Grundfunktionalität, für den speziellen Anwendungsfall der Technologieplanung, zu entwickeln.

Mit der Initialisierung dieses Projektes ergab sich für den Bearbeiter zunächst die Frage, welche Eigenschaften das Planungswerkzeug besitzen muss. Die Antwort auf diese Frage liefert das Gesamtpaket der Anforderungen, die an das zu entwickelnde System gestellt werden. Um die Anforderungen genauer bestimmen zu können, wird die Frage in kleinere Teilfragen untergliedert:

- Was soll das System leisten?
- Welche Funktionen soll das System dem Nutzer bieten?
- Wie soll das System strukturiert sein?
- Wie soll die Benutzeroberfläche aussehen?
- Wie sollen die gestellten Systemanforderungen realisiert werden?

Die Abarbeitung dieser Fragen liefert eine Liste von Eigenschaften, die die Leistung des zukünftigen Systems beschreiben, jedoch nicht unbedingt komplett definiert sein muss. Die unterschiedlichen Eigenschaften bzw. Anforderungen des Systems werden folgendermaßen kategorisiert:

- Anforderungen an die Realisierung des Systems
- Funktionale Anforderungen
- Anforderungen an die Benutzerschnittstelle
- Qualitative Anforderungen

Die Anforderungen sollten dabei möglichst vollständig, präzise und widerspruchfrei beschrieben werden. Die kategorisierten Anforderungssammlungen werden spezifiziert und in Form einer Anforderungsspezifikation, wie dem Pflichtenheft dargestellt.

2.2 Pflichtenheft

Das Pflichtenheft, oft auch als Produktdefinition oder Produktspezifikation bezeichnet, stellt eine detaillierte verbale Beschreibung der Anforderungen dar. SOMMERVILLE beschreibt es als „… die offizielle Aufstellung darüber, was von den Softwareentwicklern erwartet wird. Es sollte sowohl die Benutzeranforderungen an das System als auch eine detaillierte Spezifikation der Systemanforderungen enthalten“ ([13], S. 168). Je nachdem wie umfangreich die Software ausfallen soll, können bei einer Vielzahl von Anforderungen auch separate Dokumente erstellt werden.

Da das Pflichtenheft von unterschiedlichen Personengruppen verwendet wird stellt es zwangläufig einen Kompromiss dar. Die dargelegten Anforderungen sollten spezifisch für bestimmte Personengruppen, z.B. Anwendungsspezialisten oder Softwarearchitekten aber dennoch allgemein, in einer für den Kunden lesbaren Form vermittelt werden. Ein Pflichtenheft sollte immer für alle Projektbeteiligten gut lesbar sein und deshalb detailliert und trotzdem allgemein verbal formulierte Beschreibungen beinhalten.

Wie zuvor schon erwähnt, hängt der Detaillierungsgrad eines Pflichtenheftes von Art und Umfang des zu entwickelnden Systems ab. „Wenn die Anforderungen flexibler gehandhabt werden können und wenn ein interner, iterativer Ansatz zur Entwicklung verwendet wird, kann das Pflichtenheft weniger detailliert ausfallen. Mehrdeutigkeiten können dann während der Entwicklung des Systems geklärt werden“ sagt SOMMERVILLE ([13], S. 169). Wenn die Entwicklung eines Softwareproduktes an externe Unternehmen abgegeben wird, sollten kritische Systemspezifikationen sehr präzise und detailliert beschrieben werden.

Alle größeren Unternehmen und Organisationen wie IBM, das IEEE oder das US-Verteidigungsministerium besitzen eigene Standards und Formatierungen für Anforderungsdokumente. Abbildung 1 zeigt den Aufbau des Anforderungsdokuments nach IEEE/ANSI 830-1998 Standard. Es ist der am weitesten verbreitete Standard und kann als Grundlage für die Erstellung eines ausführlichen Pflichtenhefts genutzt werden. Allerdings ist er zu allgemein formuliert, um in dieser Form für Unternehmenszwecke genutzt werden zu können, bietet aber ein gutes allgemeines Gerüst, dass nach entsprechender Anpassung die Bedürfnisse eines Unternehmens erfüllt. ([13], S. 170)

1. **Einleitung**
   a) Ziel des Anforderungsdokuments
   b) Anwendungsbereich des Produkts
   c) Definitionen, Akronyme und Abkürzungen
   d) Referenzen
   e) Überblick über den Rest des Dokuments

2. **Allgemeine Beschreibung**
   a) Produktperspektive
   b) Produktfunktionen
   c) Benutzercharakteristika
   d) Allgemeine Beschränkungen
   e) Voraussetzungen und Abhängigkeiten

3. **Spezifische Anforderungen**, die funktionale, nichtfunktionale und Schnittstellenanforderungen umfassen. Dies ist offensichtlich der umfangreichste Teil des Dokuments, aber wegen der großen Unterschiede bei der Unternehmenspraxis ist es nicht angebracht, einen Standardaufbau für diesen Abschnitt zu definieren. Die Anforderungen können externe Schnittstellen dokumentieren, die Systemfunktionen und die Systemleistung beschreiben, logische Datenbankanforderungen spezifizieren sowie Entwurfseinschränkungen, wichtige Systemeigenschaften und Qualitätsmerkmale vorgeben.

4. **Anhänge**

5. **Index**

Abbildung 1: Aufbau Pflichtenheft nach IEEE 830-1998 (vgl. [13], S. 169)

Basierend auf dem obigen Ansatz für die Erstellung eines Pflichtenheftes werden nachfolgend einige Kernpunkte für das Projekt Samara spezifiziert und auszugsweise erläutert. Da zu Beginn des Projekts noch keine Anforderungen an die Benutzeroberfläche vorlagen, wird die Beschreibung des GUI an dieser Stelle vernachlässigt (siehe Abschnitt 3.3). Diese wird während des Projekts iterativ, unter Verwendung eines Prototypen angepasst. Für das komplette Pflichtenheft wird an dieser Stelle auf den Anhang A1 verwiesen.

Nicht Funktionale Anforderungen

Das System soll als Web-Applikation (Client-Server Applikationen) realisiert werden, die nur für den internen Gebrauch verwendet wird.

- es soll eine plattformunabhängige Web-Applikation entwickelt werden
- es soll ein Projektverwaltungsmechanismus eingebunden werden

Technische Produktumgebung

Das Produkt ist Client/Server-fähig.

Software:

- Server-Betriebssystem: Das Server-Betriebssystem ist plattformunabhängig. Es muss aber einen funktionsfähigen Web-Server zur Verfügung stellen, der PHP fähig ist, z.B. Wamp oder Xampp -Server.
- Client-Betriebssystem: Weil das System plattformunabhängig realisiert werden soll, ist die einzige Client-Voraussetzung, dass der Client einen funktionsfähigen Browser (Firefox, Internet Explorer) auf seiner Maschine installiert hat.

Hardware:

- Server: PC
- Client: Gerät mit Grafikbildschirm und installiertem Browser mit aktiviertem JavaScript

Orgware:

- Netzverbindung des Client-Rechners zum Server-PC

Spezielle Anforderungen an die Entwicklungsumgebung

Die Anforderungen an die Entwicklungsumgebung sollen nicht von der Produktumgebung abweichen. Auf der Entwicklungsmaschine müssen daher die gleichen Voraussetzungen bei der Entwicklung geschaffen werden, wie sie später auf Client-Seite vorherrschen d.h. gleiche Browser/-versionen und Einstellungen.

Gliederung der Teilprodukte

Es werden von Anfang an mehrere Teilprodukte geplant. Die ersten prototypischen Entwicklungen sollen den Weg zur Entwicklung der ersten Version des Systems zeigen. Die später folgenden Prototypen sollen die weitere Entwicklung in kleinen Schritten ermöglichen.

Nach einer ersten Analyse der Anforderungen wird klar, dass das Endprodukt als Web-Anwendung realisiert werden soll und nur zum internen Gebrauch zur Verfügung steht. Damit wird das Technologieplanungswerkzeug den wichtigsten Anforderungen wie Plattformunabhängigkeit, Installationsfreiheit und guter Portabilität gerecht, bringt aber auch veränderte Anforderungen an den Entwicklungsprozess mit sich, auf die im nächsten Abschnitt näher eingegangen wird.

2.3 Web-Anwendung

Mittlerweile hat das World Wide Web, kurz Web genannt, in allen Bereichen unseres täglichen Lebens Einzug gehalten. Nachhaltig beeinflusst es Wirtschaft und Industrie, den Bildungs- und Gesundheitssektor, die öffentliche Verwaltung und nicht zuletzt den Freizeitsektor. ([1], S. 1) „Der Grund für diese Omnipräsenz liegt insbesondere in der Natur des Web, die gekennzeichnet ist durch globale und permanente Verfügbarkeit sowie komfortablen und einheitlichen Zugriff auf beliebig verteilte, von jedermann erstellbare Informationen in Form von Webseiten“ ([1], S. 1; siehe auch [2], S. 69-77).

In den letzten Jahren durchlief das Web, welches eigentlich als reines Informationsmedium konzipiert war, eine Wandlung hin zum Anwendungsmedium. Mittlerweile haben sich Websites oder gar Web-Anwendungen zu vollwertigen, komplexen Softwaresystemen entwickelt ([1], S. 1). Verschiedene Endgeräte (z.B. PC, Fernseher, Mobiltelefon) stellen interaktive, datenintensive und personalisierte Dienste zur Verfügung, die zustandsbasiert Nutzerinteraktionen realisieren und die verwendeten Daten an im Hintergrund laufende Datenbanken übergeben ([3], S. 225-235). Web-Anwendungen unterliegen sowohl als Entwicklungsplattform, als auch als Nutzungsplattform, im Vergleich zu traditionellen Softwareanwendungen Technologien und Standards des Web und können daher wie folgt definiert werden:

„Eine Web-Anwendung ist ein Softwaresystem, das auf Spezifikationen des World Wide Web Consortium (W3C) beruht und Web-spezifische Ressourcen wie Inhalte und Dienste bereitstellt, die über eine Benutzerschnittstelle, den Web-Browser, verwendet werden“ ([1], S. 2).

Trotz der Entwicklung des Web vom Informations- zum Anwendungsmedium haben viele Unternehmen noch nicht erkannt, dass die Entwicklung von Anwendungen mehr erfordert als reine Programmierexpertise ([1], S. 2). Oftmals werden Web-Anwendungen von Entwicklern auf spontane Art und Weise, basierend auf deren individuellen Wissen und Erfahrungen umgesetzt, man spricht dabei auch von Ad-hoc-Entwicklung. Diese Art der Entwicklung zieht allerdings teils gravierende Qualitätsmängel nach sich, da die Anwendungen meist stark technologieabhängig und fehlerbehaftet sind. Für die ingenieurmäßige Entwicklung von Web-Anwendungen ist deshalb die Auswahl einer geeigneten wissenschaftlichen Herangehensweise unbedingt notwendig. Im Bereich der Softwareentwicklung spricht man in diesem Zusammenhang

auch von einem Vorgehensmodell (siehe 3.1), zu dessen Auswahl zunächst spezifische Anforderungen von Web-Anwendungen an einen Entwicklungsprozess gestellt werden.

2.4 Anforderungen an einen Entwicklungsprozess für Web-Anwendungen

Entscheidend für die Qualität von Web-Anwendungen ist in erster Linie die erfolgreiche Ermittlung der Anforderungen. Nicht zu oft ist ein Projekt schon durch fehlende, unklare oder gar fehlerhafte Anforderungen gescheitert. Das Gleiche gilt für die Anforderungen, die eine Web-Anwendung an einen Entwicklungsprozess (Vorgehensmodell) stellt. Ohne die Betrachtung der besonderen Gegebenheiten unter denen Web-Anwendungen entwickelt werden, ist die Auswahl und Anpassung eines geeigneten Vorgehensmodells nicht möglich oder führt zu großen Problemen.

Das Requirements Engineering (RE) oder der weniger gebräuchliche deutsche Begriff Anforderungstechnik „…beschäftigt sich mit Prinzipien, Methoden und Werkzeugen zur Ermittlung, Beschreibung und Prüfung von Anforderungen. RE für Web-Anwendungen steht vor besonderen Herausforderungen. Dazu zählen nicht verfügbare Stakeholder, dynamische Randbedingungen, schwer prognostizierbare Einsatzumgebungen, die besondere Bedeutung von Qualitätsaspekten und die oft fehlende Erfahrung mit Technologien“ ([1], S. 29). Allerdings gilt es zu unterscheiden zwischen dem Requirements Engineering, welches sich mit den Anforderungen an das Produkt beschäftigt, d.h. „den Wünschen des Kunden“ (siehe Kapitel 2; vgl. [1]) und dem RE, dass die speziellen Anforderungen von Web-Anwendungen an einen Entwicklungsprozess untersucht.

Dieses Kapitel widmet sich dem RE von Web-Anwendungen, bezogen auf die Besonderheiten ihres Entwicklungsprozesses und zeigt wie traditionelle Methoden und Techniken angepasst werden müssen, um erfolgreich zu sein. Gerade bei Web-Anwendungen existieren heutzutage immer noch Defizite, was den Einsatz von Methoden und Werkzeugen betrifft. Es stellt keine Seltenheit dar, dass Anforderungen ad-hoc ermittelt und dokumentiert werden, dabei ist es gerade die Komplexität von Web-Anwendungen die ein methodisch, strukturiertes Vorgehen erfordern würde.

Dass erfolgreiche Systementwicklung abhängig von der „richtigen“ Erhebung der Anforderungen ist, zeigt die Tatsache, dass in den letzten Jahren eine Vielzahl von Standards, Vorgehensmodellen, Beschreibungssprachen und Werkzeugen entwickelt

wurden. Allerdings bereitet, trotz dieser positiven Entwicklung, die Einhaltung von Anforderungen der Softwareindustrie weiterhin massive Probleme, was unter anderem auf veraltete Entwicklungsprozesse und den Versuch konventionelle Vorgehensmodelle für die Web-Entwicklung zu nutzen, zurückzuführen ist. Untersuchungen des CUTTER CONSORTIUM im Jahr 2000 ergaben zum Beispiel, dass ca. 80% aller Projekte mit Zeitverzug zu kämpfen haben und nur 16% wirklich den Anforderungen des Kunden gerecht werden.

"For most respondents, schedule delays plagued their e-projects 79% of the time and the delivered systems met business needs only 16% of the time."[4]

Die nachfolgende Tabelle verdeutlicht das Problem, sie zeigt die Erhebungen verschiedener Monitoring-Einrichtungen über Projekterfolge der amerikanischen Softwareindustrie.

Erfolgsuntersuchungen an Softwareprojekten in den USA	Projekt erfolgreich	Projekt über Budget oder über Zeit	Projekt abgebrochen
Standish Group (1994) (www.standishgroup.com)	16%	53%	31%
Center for Project Management (1995) (www.center4pm.com)	25%	50%	25%
Standish Group (2000) (www.standishgroup.com)	28%	49%	23%
Cutter Consortium (2000) (www.cutter.com)	16%	63%	21%
Gartner Group (2000) (www.gartner.com)	24%	51%	25%

Tabelle 1: Erfolgsstatistik von Softwareprojekten in den USA (vgl. [1], S. 236)

Die Ergebnisse dieser Untersuchungen zeigen, dass nur ein Viertel aller Softwareprojekte als erfolgreich eingestuft wurden. Der Rest war entweder beträchtlich über dem festgelegten Budget, hatte massiven Zeitverzug oder wurde wegen Erfolglosigkeit abgebrochen. Auf den ersten Blick erscheinen diese Ergebnisse in einem recht negativen Licht, man muss aber bedenken, dass der Umfang von Softwareprojekten kontinuierlich steigt und der zeitliche Druck immer mehr zunimmt. Deshalb kann es durchaus als Erfolg gewertet werden, dass unter Verschärfung der Randbedingungen und der sehr schnellen technologischen Entwicklung, gerade bei Web-Anwendungen, die Erfolgsstatistik hinreichend stabil verlaufen ist. ([1], S. 237)

In der IT-Industrie herrscht Einigkeit über die Bedeutung von Anforderungen an einen Entwicklungsprozess und die damit verbundene Erreichung von Zeit-, Kosten- und Qualitätszielen. Im Gegensatz zur Entwicklung herkömmlicher Software existieren bei der Web-Entwicklung im Moment noch wenig Erfahrungen. Das zieht Probleme beim Einsatz von Vorgehensweisen, Erhebungsverfahren, Beschreibungsformen und Werkzeugen nach sich. Im nächsten Abschnitt werden deshalb Eigenschaften, die bei der Entwicklung von Web-Anwendungen von Bedeutung sind, diskutiert.

2.5 Entwicklungsbezogene Charakteristika von Web-Anwendungen

Vergleicht man den Entwicklungsprozess konventioneller Softwaresysteme mit dem von Web-Anwendungen, existieren mehr Unterschiede als Gemeinsamkeiten. Trotzdem ist die Entwicklung von Web-Anwendungen ebenso durch die notwendigen Ressourcen geprägt, dazu gehören Projektmitarbeiter, die technische Infrastruktur, der Entwicklungsprozess selbst und die Integration bestehender Lösungen. (siehe auch [1], S. 18)

2.5.1 Projektmitarbeiter

In besonderem Maße sind Entwicklungsteams von Web-Anwendungen durch *Multidisziplinität* und *Juvenilität* gekennzeichnet. Diese definieren gemeinsam mit der so genannten *Community-Entwicklung* eine völlig neue Form der Zusammenarbeit, Organisation und Kommunikation verschiedener Entwicklergruppen. Die unterschiedlichen Sichtweisen und Schwerpunkte derer können nur durch einen angepassten Entwicklungsprozess, mit entsprechendem Projektmanagement, integriert werden. ([1], S. 18)

- **Multidisziplinarität**

  Im Gegensatz zur traditionellen Softwareentwicklung braucht es bei der Entwicklung von Web-Anwendungen ein durch Vielfältigkeit geprägtes Team, das sich durch unterschiedliche Fähigkeiten und Wissen auf den einzelnen Technologiegebieten auszeichnet. POWELL charakterisiert Web-Anwendungen als eine Mischung aus Printpublishing und Softwareentwicklung, Marketing und Informatik, Kunst und Technologie ([9]). „Die Entwicklung von Web-Anwendungen sollte demnach auch als *multidisziplinäres Vorgehen* betrachtet werden. […] Neben IT-Experten, die sich für die technische Realisierung der Funktionalität des Systems verantwortlich zeichnen, sollten Hypertext-Experten und Designer für die Gestaltung von Hypertext und

Präsentationen herangezogen werden, während Domänenexperten für die Inhalte verantwortlich sind“ ([1], S. 18). Je nach Art des Produktes (z.B. E-Commerce-Anwendung, Online-Shop, virtuelle Galerie) haben die verschiedenen Disziplinen unterschiedlich viel Einfluss an der Entwicklung.

- **Juvenilität**

  Bedingt durch das geringe Alter von Web-Technologien sind auch deren Entwickler deutlich jünger und unerfahrener als traditionelle Softwareentwickler. Sie zeichnen sich meist durch Technik-Affinität und großes Interesse an neuen Technologien aus, besitzen aber Defizite, was grundlegende Kenntnisse betrifft.

- **Community-Entwicklung**

  Diese Form der Kooperation unter Entwicklern stellt einen recht neuen Ansatz der Anwendungsentwicklung dar. Sie basiert auf dem Prinzip des *Sharing* von Programmcode, d.h. Entwickler binden frei zugängliche Software (Quellcode) ganz oder teilweise in ihre Projekte ein und stellen ihre eigenen Entwicklungen im Web zur Verfügung. Diese Zusammenarbeit mit ihren ungeschriebenen Gesetzen prägt eine vollkommen neue Art der Implementierung. ([1], S. 18-19)

2.5.2 Technische Infrastruktur

Die technischen Komponenten der Web-Anwendung sind wesentlich geprägt durch die Merkmale *Inhomogenität* und *Immaturität*.

- **Inhomogenität**

  Die Funktionalität von Web-Anwendungen ist maßgeblich von zwei *Fremdkomponenten* abhängig: dem Webserver, der die Inhalte zur Verfügung stellt und dem Webbrowser, der die Inhalte darstellt. Auf den Webserver hat man einen recht großen Einfluss, da dieser meist vom Entwickler selbst aufgesetzt wird oder er zumindest Anforderungen an dessen Konfiguration stellen kann. Beim Browser hingegen sieht das anders aus, da der Entwickler keinen Einfluss auf diesen hat. Zudem wird die Entwicklung durch die Browservielfalt, deren unterschiedliche Versionen und Plug-ins zusätzlich erschwert. ([1], S. 19)

- **Immaturität**

  Durch die explosionsartige Entwicklung immer neuer Web-Technologien und den damit verbundenen häufigen Versionswechseln, kann oftmals kein richtiges

Entwicklungswissen aufgebaut werden bzw. geht verloren. Deshalb sind Web-Anwendungen „vielfach noch unreif, d.h. sie sind entweder fehlerbehaftet oder verfügen nicht über die gewünschte Funktionalität“ ([1], S. 19). Gründe dafür sind unter anderem das Web-Anwendungen in immer kleineren Intervallen auf den Markt geworfen werden (*Time-to-Market*-Druck).

2.5.3 Entwicklungsprozess

Im Gegensatz zu traditionellen Softwareentwicklungsprozessen ist dieser bei Web-Anwendungen von sehr viel Dynamik geprägt. Entwickler müssen wesentlich flexibler vorgehen und viele Tätigkeiten werden parallel durchgeführt.

- **Flexibilität**

  Flexibilität spielt eine große Rolle bei der Entwicklung von Web-Anwendungen, da im Gegensatz zu sequentiellen, phasenbezogenen Entwicklungsprozessen hier nicht die Erreichung fester „Produkte“ (Ziel einer Phase), sondern die Reaktion auf sich ändernde Rahmenbedingungen, wie das Einhalten von Terminen mit flexiblen Inhalten im Vordergrund steht. ([1], S. 19)

- **Parallelität**

  Immer kürzere Entwicklungszyklen, die Notwendigkeit im Web präsent zu sein und auf Grund der „… Tatsache, dass Web-Anwendungen oftmals in autonome Komponenten strukturiert werden können (Authentifizierung, Suchfunktion, Newsticker etc.), werden viele Web-Anwendungen parallel von Subgruppen entwickelt“ ([1], S. 19-20). Die Projektmitarbeiter werden daher nicht nach ihrer Qualifikation eingeteilt, sondern entsprechend der Komponente (Teilanwendung) die sie bearbeiten. Ein ähnliches Vorgehen hat sich bei den verschiedenen Versionen einer Web-Anwendung etabliert, während z.B. die Qualitätssicherung an der aktuellen Version vorgenommen wird, wird die nächste Version bereits implementiert und die übernächste befindet sich parallel dazu schon in der Designphase. ([1], S. 19) Diese Art der parallelen Bearbeitung stellt hohe Ansprüche an das Personalmanagement, um einen reibungslosen Projektverlauf zu gewährleisten.

2.5.4 Integration

Mit dem Aufkommen neuer Technologien im Bereich der Web-Anwendungen hat die Integration enorm an Bedeutung gewonnen. Dabei bezieht sie sich mittlerweile nicht nur auf technische Aspekte, sondern schließt auch organisatorische und inhaltliche Punkte mit ein. Kaum eine Neuentwicklung kommt heute ohne Integration aus, egal ob sie interner oder externer Art ist.

- **Interne Integration**

  Oft besteht die Notwendigkeit bestehende Systeme mit ihren Inhalten in eine Web-Anwendung zu integrieren bzw. die existierenden Inhalte dieser (Legacy-) Systeme zur Verfügung zu stellen. Beispielsweise können Datenbankinhalte wie Produktkataloge, Dienstleistungsübersichten etc. in Web-Anwendungen integriert und somit potentiellen Kunden zur Verfügung gestellt werden. ([1], S. 20)

- **Externe Integration**

  Die Entwicklung von Web-Anwendungen ohne die Integration externer Inhalte und Dienste ist heutzutage fast undenkbar. Dabei handelt es sich meist um eine große Anzahl von Quellen „…die häufig wechseln und durch hohe Autonomie hinsichtlich Verfügbarkeit und Änderung der Schemata gekennzeichnet sind“ ([1], S. 20). Da diese Quellen meist nur unzureichend dokumentiert sind, liegen oftmals wenig Informationen über deren Eigenschaften, wie z.B. Inhalt oder Funktionen, vor ([1], S. 20). Bei der Kombination unterschiedlicher Web-Services sollten deshalb ausführliche Funktionstests durchgeführt werden, um evtl. auftretenden Seiteneffekten vorzubeugen.

Die in diesem Abschnitt definierten Charakteristika von Web-Anwendungen ermöglichen unter Beachtung dieser, die Auswahl und Anpassung eines geeigneten Vorgehensmodells für deren Entwicklungsprozess. Das nächste Kapitel befasst sich thematisch mit den Grundlagen, Arten und dem Entscheidungsprozess für ein Modell.

3 KONZEPTION EINER TECHNOLOGIEPLANUNGSANWENDUNG FÜR DAS PROJEKT SAMARA

3.1 Vorgehensmodelle der Softwareentwicklung und deren Ableitung für Web-Anwendungen

Allgemein ist es Aufgabe eines Vorgehensmodells, einen Prozess in strukturierte Phasen zu untergliedern und alle auftretenden Aufgabenstellungen und Aktivitäten in einer sinnfälligen logischen Ordnung darzustellen. BALZERT definiert ein solches Modell folgendermaßen:

„Jede Software-Entwicklung soll in einem festgelegten organisatorischen Rahmen erfolgen. Ein Prozess-Modell – auch Vorgehensmodell genannt – beschreibt einen solchen Rahmen. In ihm wird festgelegt, welche Aktivitäten in welcher Reihenfolge von welchen Personen erledigt werden und welche Ergebnisse – im Folgenden Artefakte (Produkte) genannt – dabei entstehen und wie diese in der Qualitätssicherung überprüft werden“ ([5], S. 54).

Die einzelnen Phasen des Prozesses werden entsprechend ihrer Funktion mit Methoden und Techniken hinterlegt. Um den komplexen Prozess der Softwareentwicklung beherrschbar zu machen und übersichtlicher zu gestalten, bedient man sich solcher Vorgehensmodelle und überführt diese in eine für das Web-Engineering nutzbare Form.

Bedingt durch das noch recht junge Alter von Web-Anwendungen sind bisher noch keine eigenen Prozesse für deren Entwicklung entstanden bzw. sind gerade am entstehen. Stattdessen versucht man für traditionelle Software konzipierte Prozesse anzupassen (siehe [7]).

Der Entwicklungsprozess durchläuft überschaubare, zeitlich und inhaltlich begrenzte Phasen und spiegelt den schrittweisen Fertigstellungsvorgang der Software wieder. Begleitet wird dieser Vorgang dabei immer vom Projektmanagement und der Qualitätssicherung.

Trotz der zahlreichen Variationen von Vorgehensmodellen kann man deren Aufbau auf die folgenden grundlegenden Phasen reduzieren:

- Problemanalyse und Planung
- Systemspezifikation (Anforderungsdefinition)
- System- und Komponentenentwurf
- Systemtest
- Betrieb und Wartung ([12], S. 17)

Grundsätzlich ist es möglich Vorgehensmodelle an Hand ihres Detailierungsgrades und ihrer Durchlaufeigenschaften differenzieren. Man unterscheidet zwischen dem einmaligen Durchlaufen einer Phase (z.B. Software-Life-Cycle-Modell) und dem mehrmaligen iterativen Durchlauf (z.B. Extreme Programming), wobei hier geringe Modifikationen einfließen können.

In der Fachwelt herrscht bisher Uneinigkeit über das optimale Modell und bis zum heutigen Zeitpunkt konnte sich keines der entwickelten Modelle durchsetzen. Die Gründe sollen in dieser Arbeit nicht weiter erörtert werden, ergeben sich aber mit hoher Wahrscheinlichkeit aus den an der Softwareentwicklung beteiligten Tätigkeitsgruppen. Dabei handelt es sich zum einen, um die von der programmiertechnischen Realisierung unabhängige Analyse von Geschäftsprozessen (Geschäftsprozessmodell und Datenmodell) zum anderen, um die EDV-technische Realisierung (Design und Programmierung). Daraus resultierend wird jede dieser Gruppen, die für sie „richtigen“ Vorgehensmodelle favorisieren.

Basierend auf ihren Eigenschaften lassen sich Vorgehensmodelle in folgende Kategorien einteilen (vgl. [8]):

- **Universelle Vorgehensmodelle** dienen der Steuerung einer Softwareentwicklung von der Konzeption bis zum Einsatz im Echtbetrieb, inklusive der im Echtzeitbetrieb anfallenden Änderungen einer Software. Man kann diese Modelle einteilen in starre Phasenabfolgen wie z.B. beim Wasserfallmodell und iterative Modelle wie z.B. das Spiralmodell.

  Vertreter: - Wasserfallmodell
  - Spiralmodell
  - Prototypische Vorgehensmodelle

- **Software-Lebenszyklusmanagement-Modelle** erweitern die Phasen über den kompletten Lebenszyklus einer Software. Dieser Typ ist eine Mischung aus Ist-Beschreibung und normativer Vorgabe. Je nach Standardisierungsgrad werden verschiedene Entwicklungsstufen vergeben. Unternehmen können sich diese Entwicklungsstufen von externen Quellen zertifizieren lassen.

  Vertreter: - Norm ISO/IEC 12207
  - Capability Maturity Model (CMM)
  - V-Modell
  - Euromethod

- **Weltliche Vorgehensmodelle** entsprechen meist einer Programmierer-Philosophie, einen bestimmten Ansatz, wie Software nach Ansicht der Proponenten, d.h. der Entwickler/Programmierer, am besten entwickelt werden sollte. Diese Philosophien beinhalten sehr oft auch Prozesselemente und werden daher ebenfalls als Prozessmodell bezeichnet.

  Vertreter: - Extreme Programming (XP)
  - Rational Unified Process (RUP)
  - V-Modell XT

3.1.1 Klassisches sequentielles Software-Life-Cycle Modell

Unter dem Begriff *Software-Life-Cycle* versteht man die Zeitspanne, in der ein Softwareprodukt entwickelt und eingesetzt wird, bis zum Ende seiner Benutzung. Hinzu kommt die Vorstellung dass dieser Zeitraum strukturiert in Phasen, denen definierte Aktivitäten und Ergebnisse zugeteilt sind, gegliedert wird.

Die einzelnen Phasen sind in ihrer Reihenfolge fest angeordnet und durch Beziehungen miteinander verknüpft. Da einzelne Phasen unter Umständen wiederholt ausgeführt werden spricht man auch von „Cycle“. Abbildung 1 zeigt den klassischen, sequentiellen und idealen Software-Entwicklungsprozess, den sogenannten *Software-Life-Cycle*. ([12], S. 18)

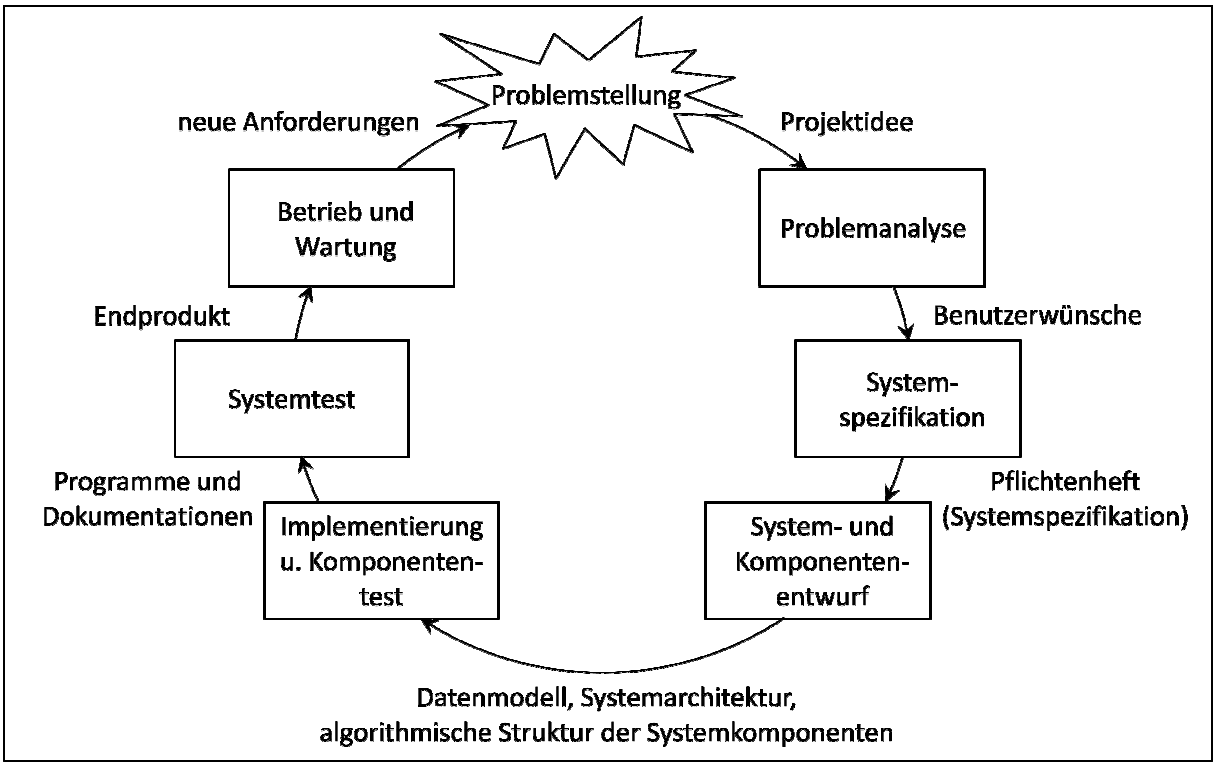

Abbildung 2: Software-Life-Cycle-Modell (nach [12], S. 18)

Um einen besseren Einblick in die einzelnen Phasen des Software-Life-Cycle, mit ihren entsprechenden Zielen, wichtigsten Tätigkeiten und Ergebnissen zu erhalten, soll auf diese hier kurz eingegangen werden, die Betrachtung der Methoden und Techniken die eingesetzt werden um diese Ziele zu erreichen, sollen an dieser Stelle nicht weiter erörtert werden. Der Fokus liegt hier zunächst auf dem Entwicklungsprozess an sich. In späteren Kapiteln dieser Arbeit werden die einzelnen Phasen des verwendeten Vorgehensmodells detailierter beschrieben und näher auf die verwendeten Techniken und Methoden eingegangen.

Problemanalyse und Planung

Inhalt der Problemanalyse und Planungsphase ist es, festzustellen und zu dokumentieren für welchen Aufgabenbereich eine Softwarelösung entwickelt werden soll. In ihr wird beschrieben welche Arbeitsschritte ausgeführt werden und welcher Art ihre Wechselwirkungen sind, welche Tätigkeiten automatisiert werden sollen und vor allem welche technischen, personellen, finanziellen und zeitlichen Ressourcen notwendig sind um das Projekt zu realisieren.

Ziel ist es das Projekt grob zu skizzieren und zu umreißen, um einen Überblick über die Bestandteile des geplanten Systems zu erhalten, sowie erste Aufwandsabschätzungen treffen zu können. Notwendigerweise ist dabei eine Erhebung des Ist-Zustands und Abgrenzung des Problembereichs unabdingbar. ([12], S. 18)

Ergebnisse: - Beschreibung des Ist-Zustands
- Projektauftrag
- grober Projektablaufplan

Systemspezifikation

In der Systemspezifikation wird festgelegt was die zu entwickelnde Software später einmal leisten soll. Dabei wird von Auftraggeber und Softwareentwickler/-hersteller definiert, welche Mindestanforderungen und Einschränkungen für die Realisierung gelten.

Die Erstellung der Anforderungsdefinition bzw. dem Pflichtenheft (Systemspezifikation) und die Ausarbeitung eines genauen Projektplanes zählen zu den wichtigsten *Tätigkeiten* dieser Phase. Die Validierung der Systemspezifikationen, wozu die Prüfung auf Vollständigkeit der Anforderungen und die Prüfung der technischen Durchführbarkeit zählen, ist ebenfalls elementarer Bestandteil. ([12], S. 19)

Ergebnisse: - Pflichtenheft (Systemspezifikation)
- genauer Projektablaufplan

System- und Komponentenentwurf

Während der Entwurfsphase wird festgelegt, welche und wie viele Komponenten erstellt werden und in welcher Weise diese, die in der Systemspezifikation definierten Anforderungen, abdecken. Außerdem wird festgelegt in welcher Weise die Systemkomponenten interagieren.

Aufgabe in dieser Phase ist es die Systemarchitektur zu entwerfen. Darunter fällt unter anderem die Definition der Systemkomponenten, der Entwurf von Schnittstellen und deren Funktionsweise, sowie Wechselwirkungen festzulegen. Falls erforderlich, sollten außerdem der Entwurf eines logischen Datenmodells, die algorithmische Struktur der

Systemkomponenten und die Validierung der Systemarchitektur, Bestandteil dieser Entwurfsphase sein. ([12], S. 19)

Ergebnisse:
- Beschreibung des log. Datenmodells
- Systemarchitektur
- algorithmische Struktur der Systemkomponenten
- Dokumentation der Entwurfsentscheidungen

Implementierung und Komponententest

Die Ergebnisse der Entwurfsphase in eine auf dem Rechner ausführbare Form zu bringen und ihre Richtigkeit zu prüfen ist das *Ziel* der Implementierungsphase.

Die Codierung, d.h. die Übertragung der Algorithmen in eine Programmiersprache und deren Verfeinerung für die einzelnen Systemkomponenten, ist Hauptaufgabe dieses Abschnitts. Daneben gilt es das logische Datenmodell in ein physisches Datenmodell (z.B. Datenbank) zu übertragen und die syntaktische Richtigkeit der Algorithmen sowie deren semantische Funktionsweise zu prüfen. ([12], S.19-20)

Ergebnisse:
- Programmcode
- Protokolle der Komponententests
- physisches Datenmodell

Systemtest

Die Systemkomponenten und deren Wechselwirkungen unter realen Bedingungen zu prüfen und möglichst viele Fehler des Softwaresystems aufzudecken, auch Bugtracking genannt, ist Ziel der Testphase. Nur so kann sichergestellt werden, dass die Systemimplementierung alle Spezifikationen erfüllt. ([12], S. 20)

Ergebnisse:
- Fehlerprotokoll

Betrieb und Wartung

Nachdem das Softwareprodukt zur Benutzung freigegeben wurde folgt die längste Phase des Software-Life-Cycles, die Betriebsphase. Ab dem Zeitpunkt der Freigabe ist es Aufgabe der Softwarewartung, die während des tatsächlichen Betriebs auftretenden Fehler zu beheben und Systemänderungen bzw. –erweiterungen durchzuführen.

Neben den eigentlichen, oben beschriebenen Projektphasen existieren noch zwei weitere Tätigkeiten, die wesentlicher Bestandteil der sequentiellen life-cycle-orientierten Software-Entwicklungsmethode sind. Die *Dokumentation* und die *Qualitätssicherung* bilden keine eigenen Phasen, da sie projektbegleitend in allen Projektphasen durchgeführt werden müssen. ([12], S. 20)

Die *Dokumentation* soll neben der Planung des Projektverlaufs und der Kalkulation der Herstellungskosten in erster Linie der Kommunikation der an dem Projekt beteiligten Personen, während der unterschiedlichen Entwicklungsphasen, dienen. ([12], S. 20) Nach Abschluss der Entwicklung hilft sie bei der Wartung des Softwareprodukts und kann ebenso Bestandteil von Schulungsunterlagen sein.

Aufgabe der *Qualitätssicherung* ist es in erster Linie dafür zu sorgen, dass das entwickelte Softwareprodukt den Spezifikationen des Pflichtenheftes hinsichtlich seiner Qualitätsmerkmale wie Korrektheit, Zuverlässigkeit, Benutzerfreundlichkeit, Wartungsfreundlichkeit, Effizienz und Portabilität entspricht. Dazu bedient sie sich analytischer, konstruktiver und organisatorischer Maßnahmen zur Qualitätsplanung, um die Erreichung der oben genannten Merkmale zu gewährleisten. (vgl. [12], S. 18-20)

Vorgehensweise bei der life-cycle-orientierten Software-Entwicklung

Bei Software, deren Entwicklung sich an dem sequentiellen Life-Cycle-Modell orientiert, kommt die sog. Top-Down-Modellierungsstrategie, auch „Topdown-Dekomposition“ genannt, zum Einsatz ([12], S.20). Dabei trifft man zunächst die Annahme, dass das ganze System aus sog. „Black Boxes“ besteht. Diese werden im Laufe des Modellierungsvorgangs schrittweise verfeinert und strukturiert. Dieser Vorgang wird solange wiederholt bis jeder Komponente des Systems eine feste Struktur zugewiesen und ihre Funktionen eindeutig beschrieben wurden. (siehe Abbildung 3)

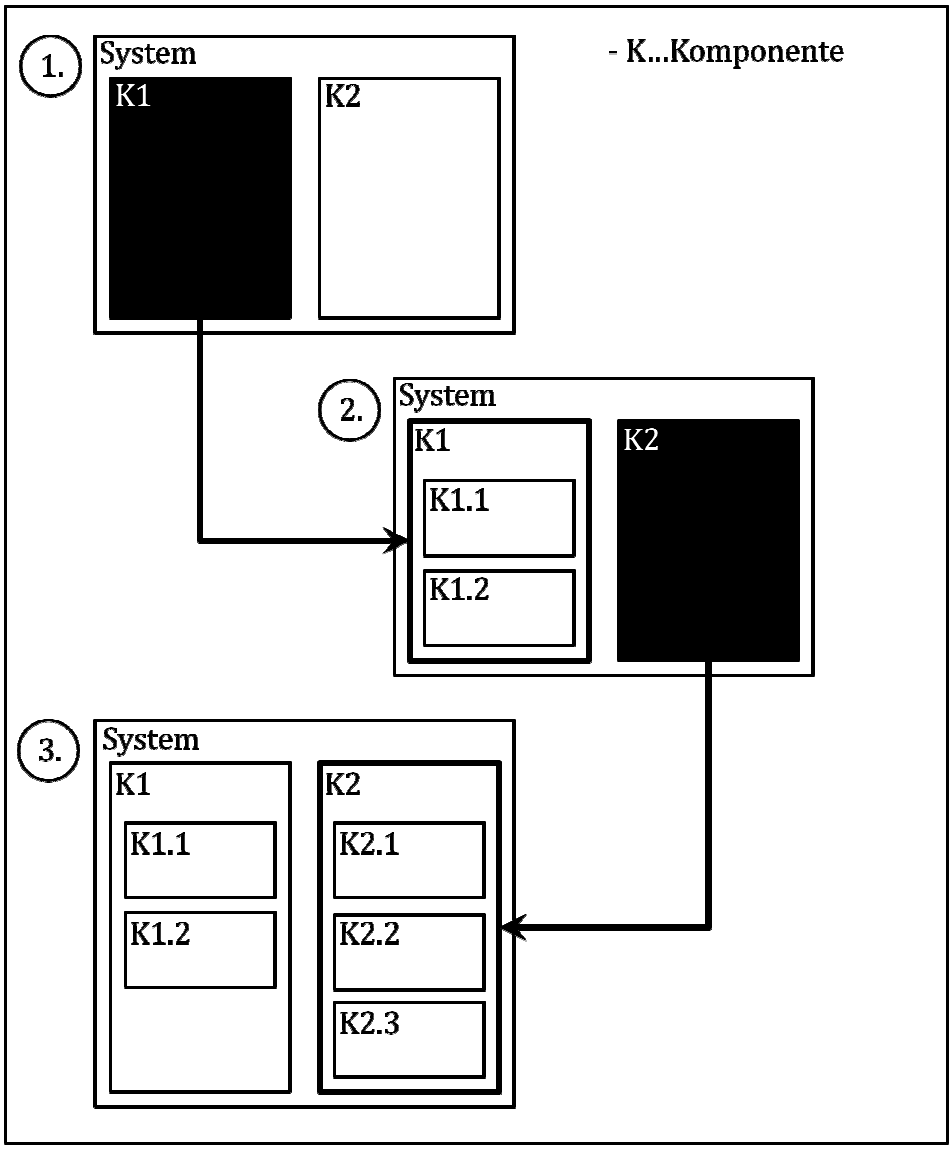

Abbildung 3: Schrittweise Konkretisierung nach dem „Topdown" – Prinzip

Wie bei allen Phasenmodellen üblich, sind auch beim Software-Life-Cycle-Modell die einzelnen Phasen klar gegeneinander abgegrenzt und sollten erst dann verlassen werden, wenn alle Ergebnisse einer Phase mittels Verifikations-/Validierungsschritt akzeptiert worden sind. Grundlage für die Verifikation eines „Phasenergebnisses" (z.B. Schnittstellendefinition) bildet die Systemspezifikation (Pflichtenheft oder Anforderungsdefinition), während im Validierungsschritt die Akzeptanz des derzeit vorliegenden Softwareprodukts hinsichtlich der Kundenwünsche geprüft wird (evtl. Korrekturen der Spezifikation bei Ablehnung). ([12], S.21)

Die *Phasenergebnisse,* welche auch als „Produkte" bezeichnet, werden liegen meist in Form von Dokumenten vor, die mit Hilfe „phasentypischer" Methoden und Werkzeuge erarbeitet werden, die die Ausgangsbasis für die nachfolgende Phase bilden.

Typisch für die Vorgehensweise nach dem Software-Life-Cycle-Modell ist die Betrachtung des Systems zunächst „von außen“. In der Analyse und Spezifikationsphase wird das System zunächst als „Black Box“ angesehen und deren Leistung definiert. Dabei ist es zunächst unerheblich wie diese Leistung einmal erbracht werden soll. Damit hat man die Wirkung des Systems nach außen genau festgelegt, aber die innere Struktur nicht weiter definiert. ([12], S. 21)

Diese Verfahrensweise kommt auch in der Entwurfsphase zum Einsatz, dabei wird zunächst die Struktur spezifiziert d.h. es wird festgelegt in welche Komponenten das System zerlegt wird, was diese leisten und wie sie interagieren (Schnittstellendefinition). An dieser Stelle wird jedoch keine Aussage über ihre tatsächliche Realisierung getroffen, dies erfolgt dann im nächsten Schritt mit dem Entwurf der algorithmischen Struktur der Systemkomponenten.

Jeder Komponente wird dabei eine entsprechende Spezifikation zugewiesen. Durch das sukzessive Zerlegen wird die Komplexität des Entwicklungsprozesses als auch des Produkts selbst gemindert. Folglich werden die Algorithmen der einzelnen Systemkomponenten auch für sich codiert und getestet. Abschließend erfolgt die Synthese der Komponenten zum Gesamtsystem, dass dann noch verifiziert werden muss. ([12], S. 22)

Das Software-Life-Cycle-Modell ist eine der am weitesten verbreiteten Entwicklungsmethoden und auch heute noch orientieren sich viele an diesem Vorgehensmodell. Aber gerade deswegen werden in der Praxis auch immer wieder Grenzen und Schwächen dieses Verfahrens aufgezeigt. Die abschließende Beurteilung soll verdeutlichen, weshalb dieses Modell für die Web-Entwicklung nicht anwendbar ist.

Vorteile:

- Es stellt einen festen Rahmen zur Verfügung, der die einzelnen Phasen des Entwicklungsprozesses klar voneinander abgrenzt und so die Projektplanung und die Abschätzung der Entwicklungskosten stark vereinfacht.
- Unabhängig von Größe, Komplexität und Anwendungsgebiet lässt sich dieses Vorgehensmodell für jedes Software-Produkt einsetzen.

- Die klare Strukturierung des Entwicklungsprozesses, vor allem während der Entwurfsphase, ermöglicht die Strukturierung des Produkts, den heute anerkannten softwaretechnischen Prinzipien anzupassen.
- Große Projekte profitieren von der sequentiellen life-cycle-orientierten Vorgehensweise, da damit ein arbeitsteiliger Entwicklungsprozess einfach realisierbar wird. ([12], S. 21)

Nachteile:

- Die Annahme, dass der Entwicklungsprozess in der Regel sequentiell ausgeführt wird und Iterationen eine Ausnahme darstellen, hat sich als falsch herausgestellt. Iterationen werden in der life-cycle-orientierten Vorgehensweise zwar mit einbezogen, allerdings existieren keine Kriterien für deren Anwendung.
- Die klare Struktur und die phasenbasierte Entwicklung schreibt vor, dass mit einer Phase erst begonnen werden darf, wenn die vorhergehende vollständig abgeschlossen ist. Realistisch gesehen ist dies aber nicht möglich, da das bedeuten würde das z.B. die Systemarchitektur von Anfang an richtig ist. Laut POMBERGER/BLASCHEK ist der Abschluss einer Phase oftmals nur dann möglich wenn nachfolgende bereits durchlaufen wurden.

  „Oft fördern erst nachgelagerte Phasen jene Informationen zutage, um eine Phase vollständig abzuschließen.“([12], S. 22)

- Die Wechselwirkungen und die überlappenden Tätigkeiten der einzelnen Phasen sind in Wirklichkeit viel komplexer als bei dieser Vorgehensweise angenommen. Die starke Trennung der einzelnen Phasen wie im sequentiellen Modell dargestellt ist eine Illusion, die nicht umsetzbar ist.
- Bedingt durch die streng sequentielle Vorgehensweise bei der Entwicklung liegen erst sehr spät im Prozess „ausführbare“ Ergebnisse vor. Beim Validierungsprozess ist man aber schon früh auf realitätsnahe Experimente angewiesen, die aber bei dieser Methode verhindert werden. Deshalb kann auch erst sehr spät auf Änderungswünsche reagiert werden. Diese können dann oftmals nur kostenintensiv und unter großem Aufwand berücksichtigt werden. ([12], S. 23)

Sicherlich stellt das Software-Life-Cycle-Modell eine wichtige Grundlage für das ingenieurtechnische Vorgehen dar, allerdings mit erheblichen Nachteilen. Die praktische Anwendung diese Modells hat gezeigt, dass es manchmal unumgänglich ist von dieser idealisierten, streng sequentiellen Vorgehensweise abzugehen, da die Definition des Zielsystems nicht völlig entbunden vom späteren Lösungsweg erfolgen kann. Für das Projekt Samara wurde deswegen eine Weiterentwicklung des Software-Life-Cycle-Modells gewählt, dass prototypingorientierte Life-Cycle-Modell, welches im nächsten Abschnitt dieser Arbeit beschrieben wird.

3.1.2 Prototypingorientiertes Life-Cycle-Modell

Die Idee prototyporientiert zu entwickeln ergab sich aus dem Ansatz heraus, das Anforderungen zu Beginn eines Projektes meist nur oberflächlich formuliert werden können, da der Kunde selbst noch keine klare Vorstellung vom Endprodukt hat. Mit dem Einsatz eines Prototypen kann sich der Anwender schon zu einem frühen Zeitpunkt der Projektlaufzeit ein Bild vom späteren Aussehen des Produkts machen. Der Kunde hat somit die Möglichkeit an Hand des Prototyps seine Ideen, Vorstellungen und Änderungswünsche einzubringen, somit verringert sich das Risiko, dass das Endprodukt nicht den Vorstellungen des Auftraggebers entspricht.

Die prototypingorientierte Entwicklungsmethode nach POMBERGER wird im Vergleich zur konventionellen Entwicklungmethode nach dem sequentiellen Life-Cycle-Modell nicht nur als iterativ angesehen, sondern schreibt bei der Vorgangsweise Iterationen explizit vor. Zwar bleibt die Phaseneinteilung des konventionellen Life-Cycle-Modells erhalten, jedoch überlappen sich Problemanalyse und Spezifikation zeitlich sehr stark. Außerdem kommt es zur Verschmelzung von Entwurfs-, Implementierungs- und Testphase. Damit sind die Phasen nicht mehr Abschnitte der stetigen Entwicklung und werden deshalb in diesem Modell als Aktivitäten bezeichnet. ([12], S. 25)

In Abbildung 4 ist das Vorgehen bei der Entwicklung nach der prototyporientierten Methodik dargestellt. Dabei wurden die parallel ablaufenden Aktivitäten überlappend dargestellt und Iterationen mit Hilfe von Pfeilen gekennzeichnet. Der Ablauf ähnelt dem klassischen Life-Cycle-Modell mit dem wesentlichen Unterschied, dass beim prototypischen Entwicklungsprozess, User-Interface-Prototypen bzw. Architektur- und Komponentenprototypen zum Einsatz kommen. Da die einzelnen Aktivitäten denen des

klassisch sequentiellen Software-Life-Cycle-Modells entsprechen, kann auf deren explizite Beschreibung an dieser Stelle verzichtet werden (siehe Abschnitt 3.1.1).

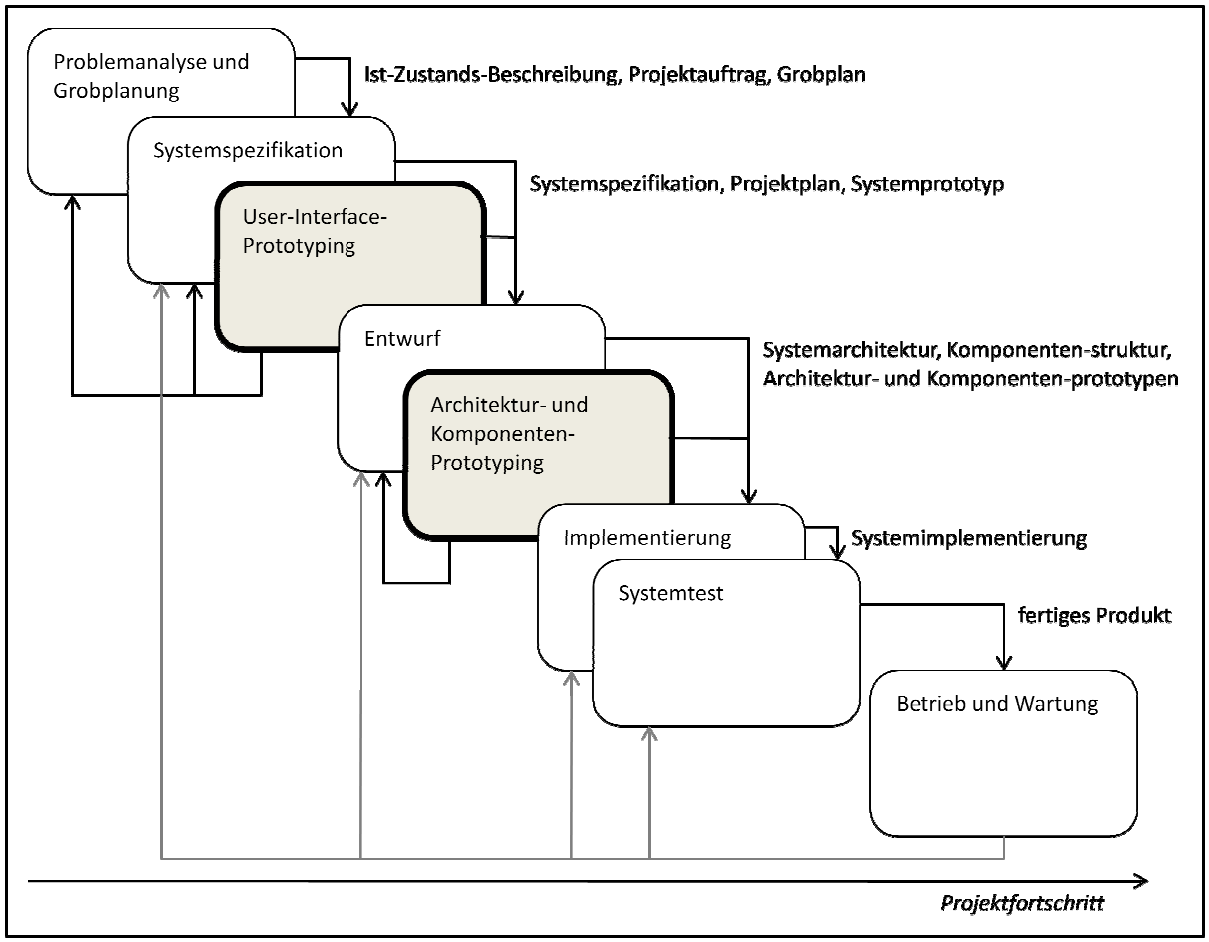

Abbildung 4: Prototyping orientiertes Life-Cycle-Modell nach POMBERGER (vgl. [12], S. 25)

Die Herstellung von Prototypen stellt einen iterativen Prozess dar, der in Abbildung 5 veranschaulicht wird. An Hand der Anforderungsspezifikation entsteht ein Prototyp mit bestimmten Eigenschaften. Dieser wird im nächsten Schritt Experimenten unterzogen, die unter realen Einsatzbedingungen ablaufen. Dabei wird geprüft, ob der Prototyp alle Anforderungen erfüllt und den Erwartungen der Anwender gerecht wird. Je nachdem ob die Web-Anwendung bzw. der Prototyp von den Testern akzeptiert wird oder nicht, werden Änderungen und Erweiterungen an der Anforderungsspezifikation vorgenommen. Dieser Prozess wird solange wiederholt bis ein Prototyp alle Anforderungen des Pflichtenheftes erfüllt. ([12], S. 25)

Wie zu Beginn dieses Abschnitts bereits angesprochen, bestehen die Ziele der prototyporientierten Software-Entwicklung in der „...Risikominimierung, einer erfolgreichen Qualitätssicherung und der Ausnutzung des Lerneffekts durch

Experimente unter realen Bedingungen“ ([12], S. 26). Im Grunde ist es unerheblich ob dabei Wegwerf- oder wiederverwendbare Prototypen genutzt werden. Im Hinblick auf die Kosten des Entwicklungsprozesses ist es allerdings ratsam Prototypen zu nutzen, die wiederverwendet werden können, um eine evolutionäre Entwicklungsstrategie zu verfolgen.

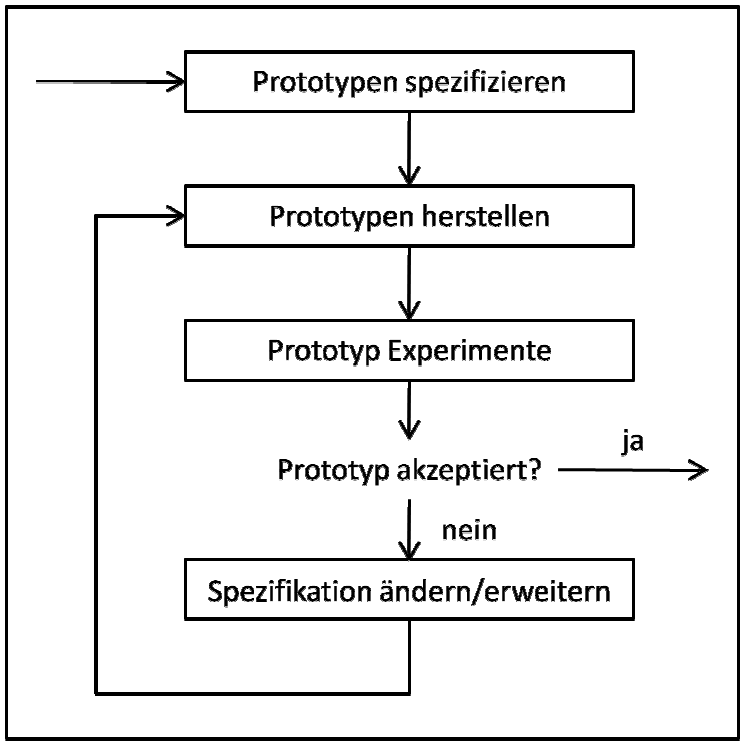

Abbildung 5: Ablauf bei der Prototyperstellung

Mit der Definition der Anforderungen in Form eines Pflichtenheftes wurde klar, dass für das Technologieplanungswerkzeug die Entscheidung für eine prototyporientierte Entwicklungsmethode enorme Vorteile mit sich bringen würde. Erstens, herrschte zu Beginn der Entwicklung noch keine Klarheit über das letztendliche Aussehen der Benutzeroberfläche. Diese kann bei der Verwendung eines Prototypen nach und nach angepasst und verändert werden. Zweitens, erfolgte die Entwicklung in unmittelbarer Nähe zu den späteren Anwendern, die auch die Rolle der Tester übernehmen sollten und mit einer frühen Implementierung zeitig in den Entwicklungsprozess einbezogen wurden. Die Entscheidung für eine prototypingorientierte Software-Entwicklungsstrategie war damit gefallen und wurde mit Hilfe des Spiralmodells in leicht abgewandelter Form für die Entwicklung der Planungsanwendung eingesetzt.

3.1.3 Spiralmodell

Bereits 1988 hat BOEHM ein Software Entwicklungsmodell vorgeschlagen das sich mit anderen Entwicklungsmodellen kombinieren lässt bzw. deren Prozesse integriert. Damit bietet es die Möglichkeit, je nach Art des Projekts die optimale Vorgangsweise zu wählen.

Anstatt wie bisher die einzelnen Phasen oder Aktivitäten als lineare Folge mit Rückwärtsbezügen darzustellen, wird der Entwicklungsprozess hier als Spirale veranschaulicht ([13], S. 103). Abgeleitet von seiner Darstellungsform wird es als Spiralmodell bezeichnet. Wie aus Abbildung 6 ersichtlich, stellt die radiale Ausdehnung den Gesamtaufwand zu einem bestimmten Zeitpunkt dar, während die Winkeldimension den aktuellen Bearbeitungsschritt angibt ([12], S. 27).

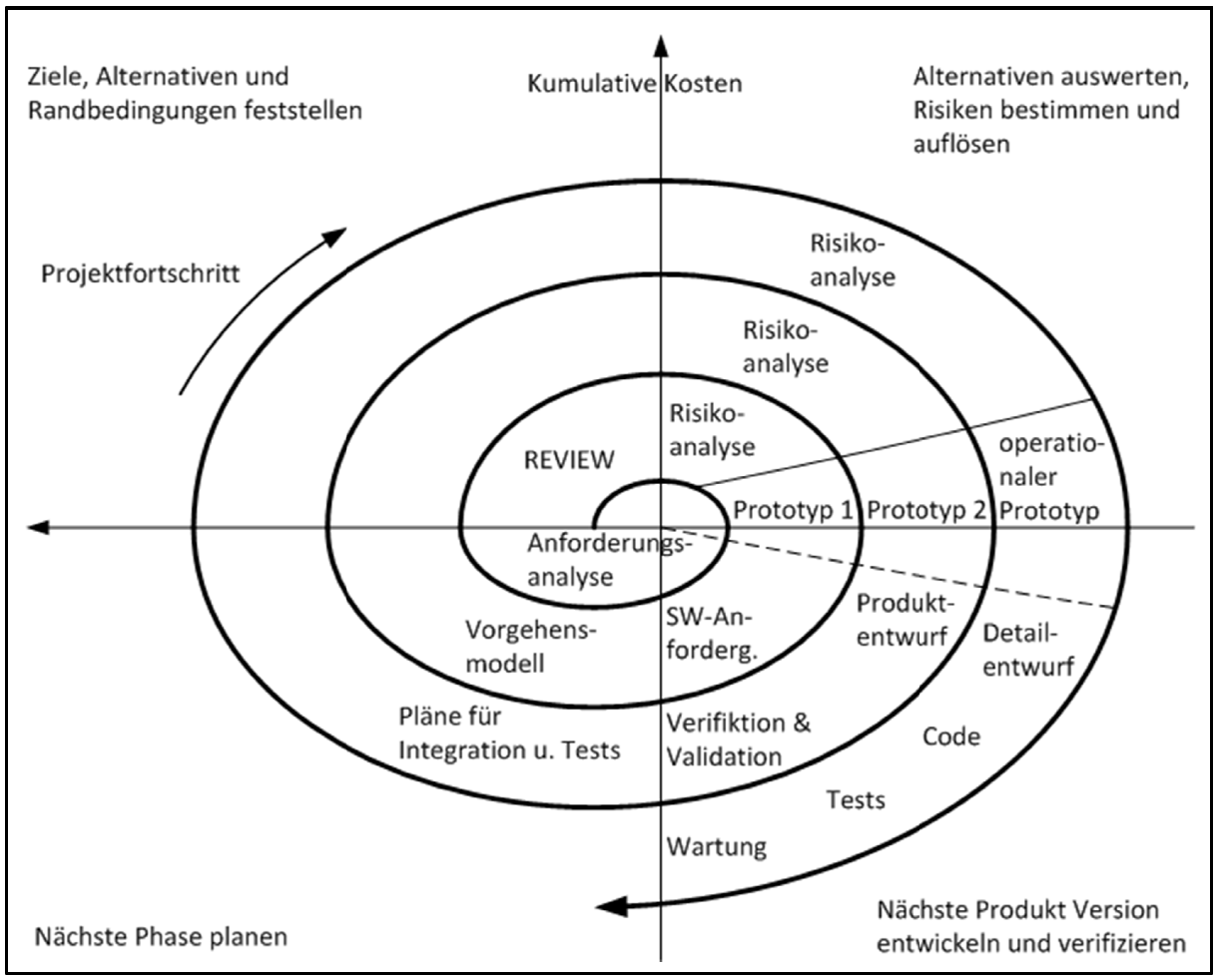

Abbildung 6: Spiralmodell nach BOEHM

Die Windungen der Spirale repräsentieren dabei die einzelnen Phasen des Prozesses. Während die innerste Windung sich mit der Anforderungsanalyse beschäftigt, hat die

nächste den Entwurf zur Aufgabe. So geht es weiter bis zum fertigen Produkt. ([13], S. 103)
Jeder Zyklus der Spirale durchläuft dabei die folgenden Segmente:

- *Ziele aufstellen:* An dieser Stelle werden Ziele und Randbedingungen, denen die Produkte unterliegen, für die aktuelle Phase des Projekts definiert. Dazu werden ein Managementplan und eine Liste der Projektrisiken erstellt.
- *Risiken einschätzen und verringern:* Die aufgelisteten Projektrisiken werden analysiert und es werden Schritte unternommen um diese zu minimieren oder gar zu eliminieren, z.B. durch Einsatz eines Prototypen.
- *Entwicklung und Validierung:* Nach Abschätzung der Risiken erfolgt die Auswahl eines Entwicklungsmodells. Zum Beispiel kann bei unklaren Anforderungen an die Benutzeroberfläche, die evolutionäre Herstellung eines Prototyps zur Anwendung kommen.
- *Planung:* Der Planungsabschnitt entscheidet darüber ob in der nächsten Windung der Spirale fortgefahren werden kann. Fällt diese Entscheidung positiv aus werden Pläne für die nächste Phase aufgestellt. ([13], S. 103)

Die ausführliche Betrachtung der Risiken hat einen hohen Stellenwert beim Spiralmodell, da deren Eintreten schnell die Kosten eines Projekts sprengen könnte. Es eignet sich daher besonders gut zum Abschätzen der Entwicklungskosten eines Software-Entwicklungsprojektes.

Der Vorteil des Spiralmodells liegt in seiner Flexibilität. Es erlaubt die Integration verschiedenster Vorgehensmodelle und lässt sich somit für jede Art von Projekt entsprechend konfigurieren.

Dank seiner Vielseitigkeit, wurde es auch für die Entwicklung des Technologieplanungswerkzeugs ausgewählt, wo es in Kombination mit der evolutionären prototyporientierten Software-Life-Cycle-Methodik zum Einsatz kommt. Das nächste Kapitel behandelt die Integration dieses Vorgehensmodells und den sich daraus ergebenden Projektablauf.

3.1.4 Auswahl eines geeigneten Vorgehensmodells für den Projektablauf

Da Softwareprodukte in Laufe der Jahre an Umfang und Komplexität zugelegt haben, gestaltet sich auch deren Entwicklung komplizierter. Deshalb braucht man einen überschaubaren Projektablaufbahn, der den Entwicklungsprozess inhaltlich und zeitlich klar unterteilt.

Für die Entwicklung des Technologieplanungswerkzeugs entschied man sich deshalb für das Vorgehensmodell des „Evolutionären Prototyping“ gepaart mit dem Spiralmodell, da damit während des Projektes sowohl die Kundenzufriedenheit als auch die Projektentwicklung optimal unterstützt werden konnte. Gemäß dem Motto:

„Ein Bild sagt mehr als tausend Worte, ein Prototyp mehr als tausend Bilder“,

ist das Ziel des „Evolutionären Prototyping“ eine evolutionär bedingte inkrementelle Softwareentwicklung, d.h. eine schrittweise aufeinander aufbauende Entwicklung.

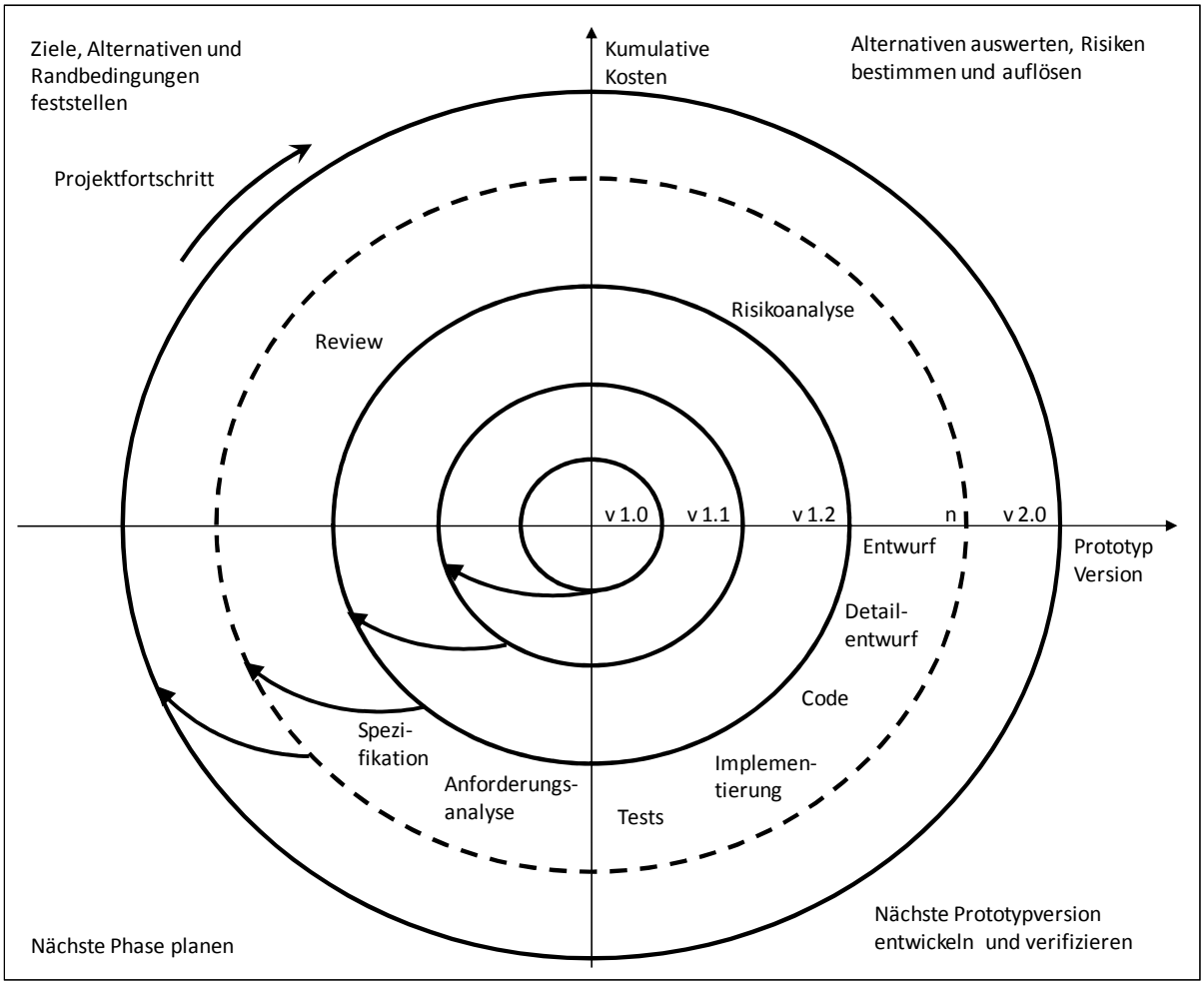

Abbildung 7: Spiralförmiges prototypisches Vorgehensmodell

Dabei entwickelt man zunächst einen Prototypen der den Basisanforderungen genügt, z.B. eine einfache Benutzeroberfläche mit Grundfunktionalität. Das Resultat wird als

Basis für die weitere Entwicklung genutzt. Dabei werden die Anforderungen an Benutzeroberfläche und Funktionalität mit jedem Schritt präziser formuliert bzw. Änderungen vorgenommen. Bei dieser Art der Vorgehensweise kann der Prototyp immer weiter verwendet werden und es werden keine unnötigen Mittel in das Erstellen von Wegwerfversionen investiert.

Mit jeder neuen Version des Prototyps springt man in der Spirale eine Windung nach außen, wobei es bei nicht Akzeptanz möglich ist, eine Windung erneut zu durchlaufen um den Prototyp anzupassen. Dabei werden immer wieder die Phasen bzw. Aktivitäten des prototypingorientierten Life-Cycle-Modells durchlaufen. Abbildung 7 veranschaulicht den Vorgang der Prototypweiterentwicklung. Die Funktionen und das gestalterische Aussehen des Prototyps werden ständig erweitert und immer weiter vervollständigt. Wichtig ist es, dass eine klare Vorstellung des Endprodukts vorliegt, damit die Entwicklung nicht ins Unendliche fortgesetzt wird, denn neue Anforderungen wird es immer geben.

Der iterative Einsatz des „Evolutionären Prototyping“ zeichnet sich durch viele Vorteile aus:

- Durch die frühe Herstellung des Prototyps kann ein besseres Verständnis zwischen Auftraggebern, Anwendern und Systementwicklern hergestellt werden.
- Die Abschätzung des Risikos und der schrittweise Aufbau des Produkts verringern das Risiko der Fehlentwicklung.
- Aktive Unterstützung der Prototyp-Entwicklung durch den Auftraggeber erhöht die Akzeptanz der Web-Anwendung.
- Die zukünftige Arbeit mit der Software fällt leichter, da bereits Kenntnisse und Erfahrungen aus der Prototyp-Entwicklungsphase vorliegen.
- Die spiralförmige Entwicklung von Version zu Version lässt eine gute Abschätzung der Entwicklungskosten zu.

Sicherlich bringt die inkrementelle Entwicklung auf Basis von Prototypen nicht nur Vorteile mit sich, die Nachteile können aber vermieden werden:

- Die Vielzahl von Prototypen kann zu einem kontinuierlich wachsenden Projekt übergehen.

- Die Methodik ist nur erfolgreich, wenn der Auftraggeber sich kontinuierlich in den Entwicklungsprozess mit einbringt.
- Eine schlechte Planung kann zu einer Mehrzahl an Prototypen führen, die den Entwicklungsprozess chaotisch gestalten.

Mit dem weiterentwickelten Spiralmodell und dem Einsatz von Prototypen wurde für diese Art und Größe von Projekt eine optimale Vorgehensweise gefunden, welche sich etabliert hat und von allen Projektbeteiligten gut aufgenommen wurde. Sicherlich hätten auch andere Vorgehensmodelle wie RUP oder XP zum Einsatz kommen können. Durch den begrenzten Projektumfang und da es sich um eine Web-Anwendung handelt, fielen diese aber aus dem Raster oder waren nicht durchführbar. Letztendlich kann festgehalten werden, dass durch den Einsatz des Vorgehensmodells der Entwicklungsprozess wesentlich strukturierter und übersichtlicher ablief, als das bei einer ad-hoc Entwicklung jemals möglich gewesen wäre. Allerdings sollte der erhöhte zeitliche und organisatorische Aufwand, der bedingt durch den Einsatz eines solchen Modells entsteht, nicht unterschätzt bzw. bei der Kalkulation berücksichtigt werden.

3.2 Architektur

Nachdem in den vorangegangenen Kapitel die Anforderungen der Web-Anwendung und die Vorgehensweise bei deren Entwicklung bestimmt wurden, steht jetzt eine neue Frage im Vordergrund: Die Architektur des Projekts.

3.2.1 Grundlagen Architektur

Für den Begriff Architektur existiert eine große Anzahl unterschiedlicher Definitionen, deshalb soll an dieser Stelle auf eine Definition verzichtet werden und nur auf die wichtigsten Merkmale nach STARKE eingegangen werden (vgl. [16]):

- *Architektur beschreibt Struktur:* Die Architektur wiederspiegelt die Struktur eines Softwaresystems inklusive ihrer Zerlegung in Komponenten. Dabei beschreibt sie statische und dynamische Aspekte des Systems und kann deshalb als Bau- und Ablaufplan dienen.
- *Architektur bildet den Übergang zwischen Analyse und Realisierung:* Die Architekturerstellung stellt einen iterativen Vorgang dar, die funktionalen Anforderungen und

die Qualitätsanforderungen auf Komponenten und deren Beziehungen und Schnittstellen zu differenzieren.

- *Architektur kann unterschiedliche Sichtweisen beinhalten:* Es ist möglich eine Architektur aus verschiedenen Blickwinkeln zu betrachten. Dabei unterscheidet man üblicherweise vier verschiedene Sichtweisen (vgl. [6], S. 42-50): Die *konzeptuelle Sicht* identifiziert die Objekte der Anwendungsdomäne und deren Beziehungen, während die *Laufzeitsicht* die Komponenten des Systems zur Laufzeit beschreibt. Hingegen beschreibt die *Prozesssicht* die Abläufe zur Laufzeit inklusive Synchronisation und Nebenläufigkeit. Die Beschreibung von Softwareartefakten sowie Subsystemen, Komponenten oder Quellcode ist Aufgabe der *Implementierungssicht*.
- *Architektur macht Systeme verständlich:* Durch die unterschiedlichen Sichtweisen werden Softwaresysteme verständlicher und komplexe Strukturen beherrschbar.
- *Architektur ist der Rahmen für flexible Systeme:* Die Architektur ist der Rahmen in dem sich ein Softwaresystem entwickeln kann. Die Erweiterung eines Systems um eine nicht in der Architektur vorgesehene Komponente ist oft kompliziert oder unmöglich. ([1], S. 79)

Diese Merkmale verdeutlichen, dass die Festlegung und Entwicklung einer geeigneten Architektur von großer Bedeutung für die Entwicklung des Softwareproduktes ist.

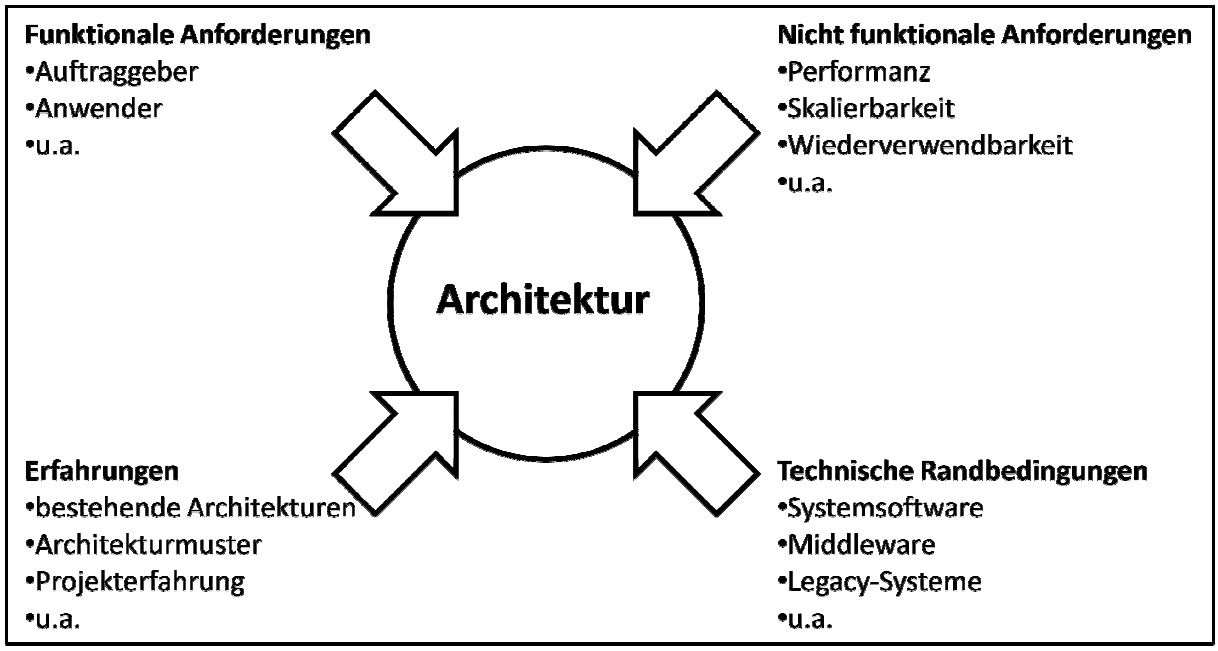

Abbildung 8: Einflussfaktoren auf die Entwicklung einer Architektur (nach [17])

Allerdings sind auch Architekturen vor, während und nach der Entwicklung Änderungen unterworfen, das ist meist „… auf unklare Anforderungen zu Beginn der Architekturentwicklung oder die Weiterentwicklung der Anforderungen nach Fertigstellung des Systems“ zurückzuführen ([1], S. 79). Abbildung 8 zeigt die verschiedenen Faktoren und ihren Einfluss auf die Architektur.

Da es sich bei Software-Architekturen um ständig ändernde Gebilde handelt, werden diese in der Fachsprache auch als „moving target’s“ bezeichnet, was die beweglichen Ziele der Architektur symbolisiert. Primär wird die Architektur durch funktionale und nicht funktionale Anforderungen (siehe Kapitel 2) beeinflusst. Daneben spielen vor allem Randbedingungen wie die Systemsoftware (z.B. das Betriebssystem), verwendete Middleware, Standards, Entwicklungsvorgaben und die Erfahrungen des Softwarearchitekten eine Rolle. ([1], S. 80)

Da komplexe Softwarearchitekturen schwierig zu entwickeln sind, werden diese in einem iterativen Vorgehen bestimmt (siehe auch Abbildung 7). Dadurch soll das Risiko, hervorgerufen durch unsichere Anforderungen und Randbedingungen, minimiert werden. Bei recht komplexen Architekturen oder wenn nur wenig Erfahrung in der Entwicklung einer Architektur vorhanden ist und der Softwareentwicklungsprozess keine Hilfestellung leisten kann, ist der Einsatz sog. *Muster* (Patterns) ratsam. Diese „… beschreiben wiederkehrende Entwurfsprobleme, welche in einem spezifischen Entwurfskontext auftreten und geben für diese Lösungsansätze an. Der Lösungsansatz beschreibt dabei die beteiligten Komponenten, deren Verantwortlichkeiten und die Beziehungen zwischen den Komponenten in der konkreten Problemstellung“ ([1], S. 81).

Entwurfswissen, welches vorher erprobt und gefestigt wurde, kommt damit bei der Entwicklung zum Einsatz und unterstützt die Herstellung qualitativ hochwertiger Softwareprodukte. Für viele unterschiedliche Arten von Software-Architekturen stehen mittlerweile solche Entwurfsmuster zur Verfügung, z.B. J2EE oder CORBA. Auch die 2-Schichten-Architektur, die Inhalt des nächsten Kapitels ist, stellt ein für Web-Anwendungen typisches Muster dar.

3.2.2 2-Schichten-Architektur des Technologieplanungswerkzeugs

Dass das Technologieplanungswerkzeug als Web-Anwendung realisiert werden soll, die nur für den internen Gebrauch bestimmt ist, bringt Vorteile hinsichtlich des Architekturlayouts mit sich. Außerdem werden Web- und Datenbankserver auf einer Maschine realisiert was die Struktur ebenfalls einfacher gestaltet, da damit die Kommunikation nur zwischen einem Server und den Clients erfolgt.

Die Kommunikation funktioniert nach dem für Web-Anwendungen üblichen Request-Response-Prinzip d.h., eine Komponente (z.B. Webbrowser) stellt eine Anfrage an eine andere Komponente (z.B. Webserver) und die Antwort auf diese Anfrage wird über denselben Kommunikationskanal zurückgesendet (synchrone Kommunikation) ([1], S. 85).

Als Grundlage für die Technologieplanungsanwendung wird die 2-Schichten-Architektur nach ANASTOPOULOS und ROMBERG verwendet, die auch als Client/Server-Architektur bezeichnet wird. In Abbildung 9 ist der Aufbau inklusive der verwendeten Komponenten dargestellt.

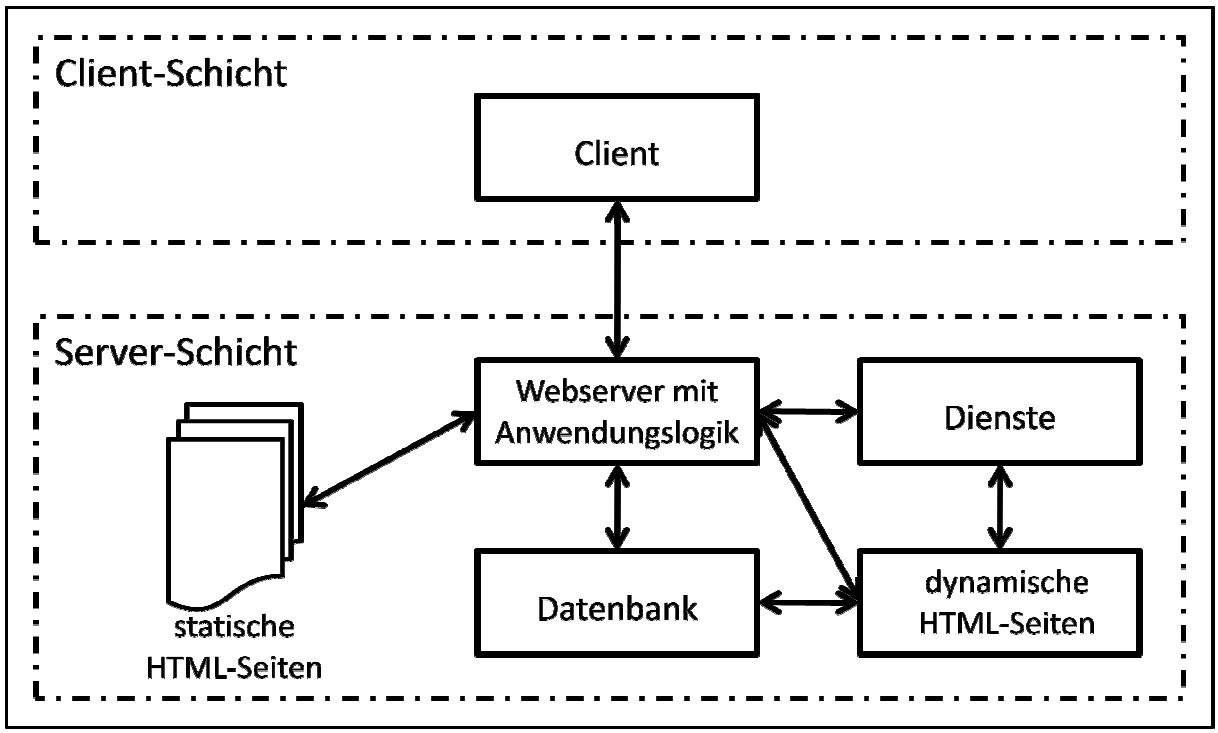

Abbildung 9: 2-Schichten-Architektur für Web-Anwendungen (nach[18])

Dem Client werden über den Webserver Dienste zur Verfügung gestellt, dabei kann die Clientanfrage direkt auf statische HTML-Seiten verweisen die keinerlei Verarbeitungslogik an der Server-Schicht erfordern. Auf Datenbanken basierende Web-Anwendungen greifen über die Logik des Webservers (z.B. CGI-Skripte) auf Datenbankinhalte zu und generieren daraus HTML-Seiten. ([1], S. 87) Bei dynamischen HTML-Seiten sind die Skriptanweisungen in den Quellcode eingebettet (z.B. MySQL-Abfragen) und werden vom Webserver interpretiert und ausgeführt.

Das Technologieplanungswerkzeug verwendet sowohl statische als auch dynamische HTML-Seiten, die je nach Clientanfrage zur Verfügung gestellt werden. Dabei werden dynamische Inhalte wie Übersichtstabellen oder Eingabeformulare mit Vorauswahl, aus einer im Hintergrund laufenden Datenbank abgefragt und mittels HTML-Seite über den Server zur Verfügung gestellt. Statische Elemente, wie Menüs, werden dagegen wie zuvor bereits erwähnt direkt bereitgestellt und erfordern keinerlei Verarbeitungsschritte des Webservers.

3.3 Grafische Benutzeroberfläche (GUI)

Nachdem im vorhergehenden Kapitel die Architektur der Technologieplanungs-anwendung definiert wurde, widmet sich dieser Abschnitt der Benutzeroberfläche. Das Graphical User Interface (GUI) wie es auch bezeichnet wird, stellt in ganz besonderem Ausmaß ein Schlüsselkriterium für die Gebrauchstauglichkeit der Anwendung dar, denn sie ermöglicht die Interaktion des Anwenders mit der Software.

Die Gestaltung der Benutzeroberfläche ist Teil der Konzeptphase, allerdings wird ein Großteil des Entwurfs bereits in der Anforderungsspezifikation erledigt, denn in diesem Dokument wird die Art und Weise der Softwarebedienung beschrieben. Da im Falle der Entwicklung des Technologieplanungswerkzeugs zu Beginn der Projektlaufzeit noch keine klare Vorstellung von Aufbau und grafischer Gestaltung der Oberfläche existierte, erstellte man zunächst ein GUI-Grobkonzept. Dieses wurde im Laufe der evolutionären Entwicklung mit Hilfe des Prototypen detailliert und vervollkommnet. Das Grobkonzept setzt sich aus den Punkten Inhaltsstruktur, Funktionalität, Seiten-Aufbau, Inhaltsformen und Benutzerführung zusammen, auf die hier eingegangen werden soll.

Inhaltsstruktur

Bei der Strukturierung der Inhalte, die auf der Web-Seite der Web-Anwendung präsentiert werden sollen, spielen zwei Ansätze eine grundlegende Rolle:

- Wie kann der Nutzer durch die Web-Anwendung navigieren?
- Wie sollen die Inhalte dargestellt werden?

Die Struktur der Web-Anwendung gibt Aufschluss über die Inhalte und wie diese miteinander verknüpft sind. Abbildung 10 zeigt die Struktur des Technologieplanungswerkzeugs und liefert damit die Antwort auf die erste Frage.

Der Inhalt lässt sich grob in fünf Bereiche untergliedern:

- Der erste Bereich beinhaltet dabei die Stammdaten, dabei handelt es sich um Produkt-, Maschinen-, Lieferanten-, Aufspannungs-, Wechselteil-, Werkzeug- und Prüfmitteldaten.
- Im Bereich zwei sind die Bearbeitungs- und Prüfoperation angeordnet. Diese bieten die Möglichkeit zum Anlegen von Prozessketten basierend auf den Stammdaten.
- Die Dokumente, die später einmal exportiert werden, sind im Bereich drei angesiedelt.
- Die Ausgabe der Dokumente d.h. die Darstellung der Prozessansicht oder des Prüfplans findet im vierten Bereich statt.
- Eine finale Auflistung der Verfügbarkeiten inklusive Prozessablaufplan wird im fünften Bereich realisiert.

Die einzelnen Bereiche sind dabei symbolisch unterscheidbar (Startseite) und werden im Menü gegeneinander abgegrenzt.

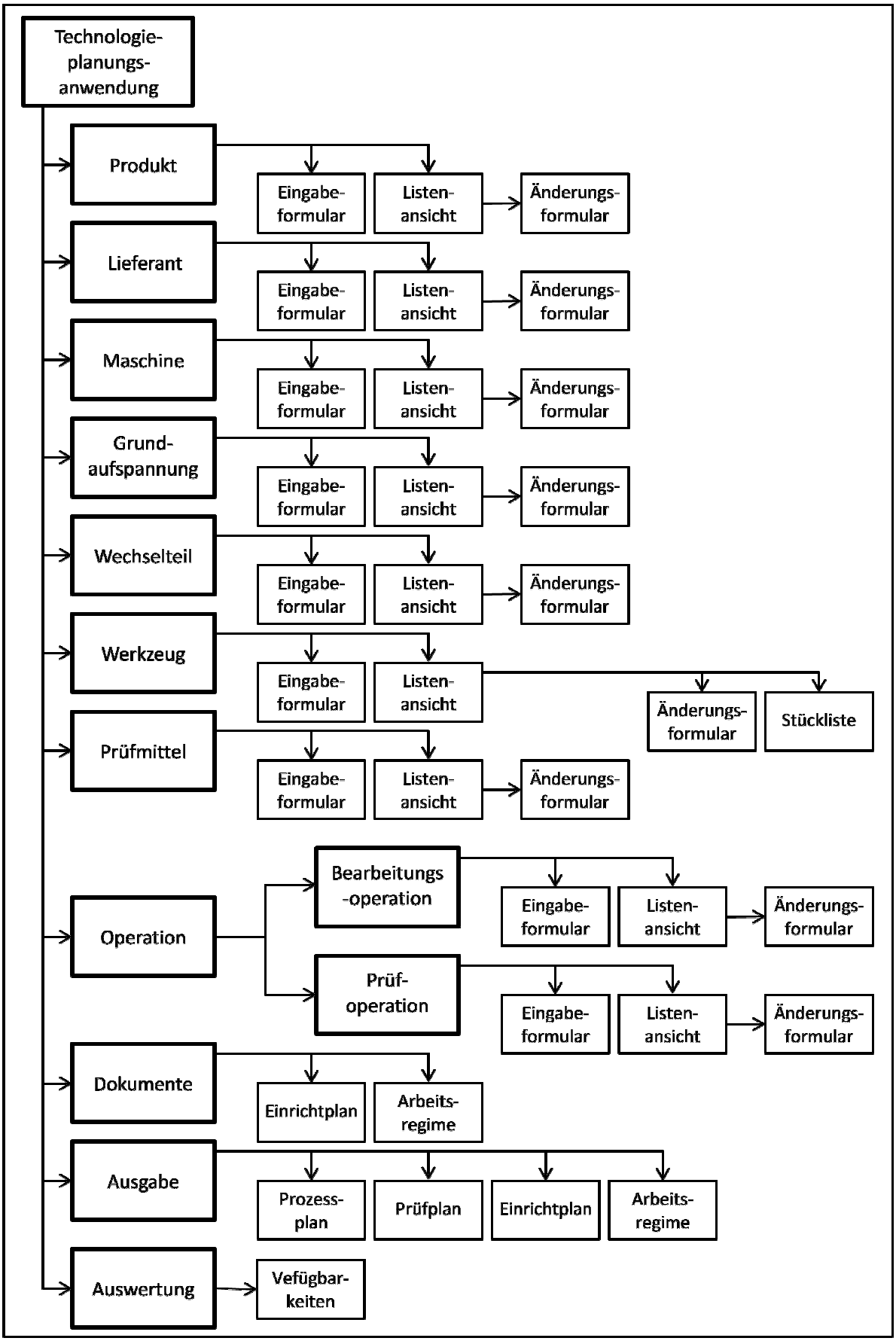

Abbildung 10: Strukturplan des Technologieplanungswerkzeugs

Funktionalität

Um dem Anwender die Orientierung zu erleichtern, ist es wichtig, dass Funktionsbereiche eindeutig definiert werden, denn „…eine durch die Nicht-Linearität der Navigation bedingte Desorientierung muss um jeden Preis vermieden werden. Dazu ist es notwendig, dem Benutzer möglichst schnell ein mentales Modell der Navigationsstruktur zu vermitteln“ ([1], S. 277). Das wird durch die kooperative Anordnung der Menüs und Funktionen erreicht. Durch die Einteilung in die verschiedenen Funktionsbereiche der Web-Anwendung, wird es ermöglicht, die bereitgestellten Funktionen übersichtlich und in einem sinnvollen Zusammenhang zu präsentieren. Die Untergliederung der Funktionsbereiche für das Technologie-planungswerkzeug wird wie folgt definiert:

- Navigations-Bereich:
  Der Navigations-Bereich ermöglicht dem Nutzer die Navigation innerhalb der Planungsanwendung. Über ein Menü kann er die von ihm gewünschte Seite mit den entsprechenden Funktionen abrufen.

- Informationsdarstellungs-Bereich:
  Der Informationsdarstellungs-Bereich dient der Darstellung der Stammdaten in Listenform. Er dient dazu sich einen Überblick zu verschaffen und bietet dazu verschiedene Filterfunktionen.

- Informationsänderungs-Bereich:
  Dieser Bereich ermöglicht neben der Darstellung eines Datensatzes auch die Änderung der einzelnen Felder eines Datensatzes.

Seiten-Aufbau

Der Seitenaufbau definiert die Positionen der unterschiedlichen Bereiche der Web-Anwendung. Bei der Aufteilung der verschiedenen Bereiche orientierte man sich am klassischen Design einer Webseite, um dem Nutzer die Bedienung und Einarbeitung zu erleichtern. Die Arbeitsoberfläche wurde dabei in drei Bereiche (Frames) unterteilt, der obere Frame (Header) verläuft dabei von links nach rechts und beinhaltet das Firmenlogo und Spracheinstellungen sowie zusätzliche Projektinformationen. Im linken Frame befindet sich der Navigations-Bereich, welcher dem Benutzer die Bewegung durch die Anwendung ermöglicht. Rechts davon im Hauptbereich werden der

Informationsdarstellungs- und der Informationsänderungsbereich untergebracht und je nach Bedarf der entsprechende Inhalt dargestellt. Die nachstehende Abbildung verdeutlicht die beschriebene Anordnung.

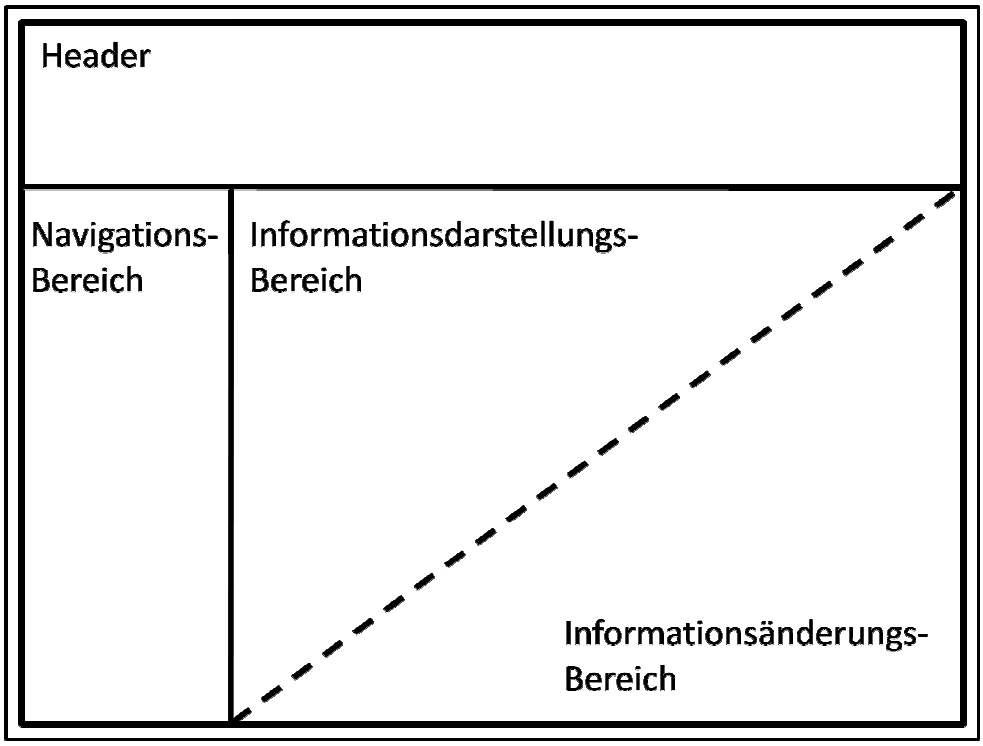

Abbildung 11: Anordnung der Funktionsbereiche

Inhaltsformen

Zur Darstellung von Inhalten existieren eine ganze Reihe verschiedener Elemente. Die Präsentation der Web-Anwendung stützt sich dabei allerdings ausschließlich auf statische Elemente und es kommen nur Texte, Grafiken und Bilder zum Einsatz. Der Nutzer hat auf diese Form des Inhalts keinen Einfluss, hat aber die Möglichkeit Dokumente und Zeichnungen seinen Datensätzen hinzu zu fügen.

- Textdarstellung: Das komplette Arsenal der Datenbestände wie Lieferanten-, Maschinen- oder Produktdaten usw. fällt in diese Kategorie.
- Grafikdarstellung: Grafiken repräsentieren in Form von Logos und Bildern Informationen auf der Web-Seite.

Die Präsentation der Inhalte ist dabei an das Design anderer firmeninterner Anwendungen angelehnt. Der Wiedererkennungswert steigt und Anwendern wird die Einarbeitung in die Anwendung erleichtert. Umgesetzt wurden die Vorgaben der Firma vor allem in der Farbgebung des Technologieplanungswerkzeugs sowie im Header-

Bereich. Im Kapitel „Umsetzung“ Unterkapitel „Design und Ergonomie“ werden die einzelnen Elemente genauer beschrieben.

Benutzerführung

Durch die verstärkte Integration des Technologieplanungs-Teams in den Entwicklungsprozess, wurden die Anwender mit der Navigation und der Informationsdarstellung vertraut gemacht. Das Testen der Prototypen stellte ein gutes Mittel zur Schulung der Mitarbeiter dar. Die Zielgruppe der Anwender beschränkt sich auf eine kleine Anzahl Benutzer, deshalb war die Erstellung eines Konzepts eigens für die Benutzerführung nicht notwendig.

3.4 Datenverwaltung

Beim Ausführen eines Programms, sind die im Programm verwendeten und erzeugten Daten nur solange vorhanden wie das Programm „läuft“. Die Daten würden deshalb nach Programmende mitsamt ihren Informationen verloren gehen. Deshalb ist es notwendig, Daten die wiederverwendet werden sollen, dauerhaft zu speichern und solange verfügbar zu machen, bis sie explizit gelöscht oder überschrieben werden.

Ein Mittel zum persistenten speichern von Informationen stellen Datenbanken (DB) dar. Diese speichern vom Entwickler definierte Datensätze zuverlässig und unabhängig vom Programm. Da die Datenbank programmunabhängig ist braucht es zur Nutzung eine Verwaltung, die die Zugriffe auf die DB steuert. Die Kombination aus Datenbank und Verwaltungs- bzw. Managementsystem wird deshalb als Datenbanksystem (DBS) bezeichnet. Mit Hilfe spezieller Anfragesprachen (z.B. MySQL) ermöglicht das DBS den Zugriff auf die in der Datenbank abgelegten Inhalte.

Nach BALZERT besteht ein Datenbanksystem aus einer oder mehreren Datenbanken (DB), einem *Data Dictionary* (DD), und einem Datenbankmanagementsystem (DBMS) ([5], S. 721). Die nachfolgende Abbildung verdeutlicht den grundsätzlichen Aufbau des Systems.

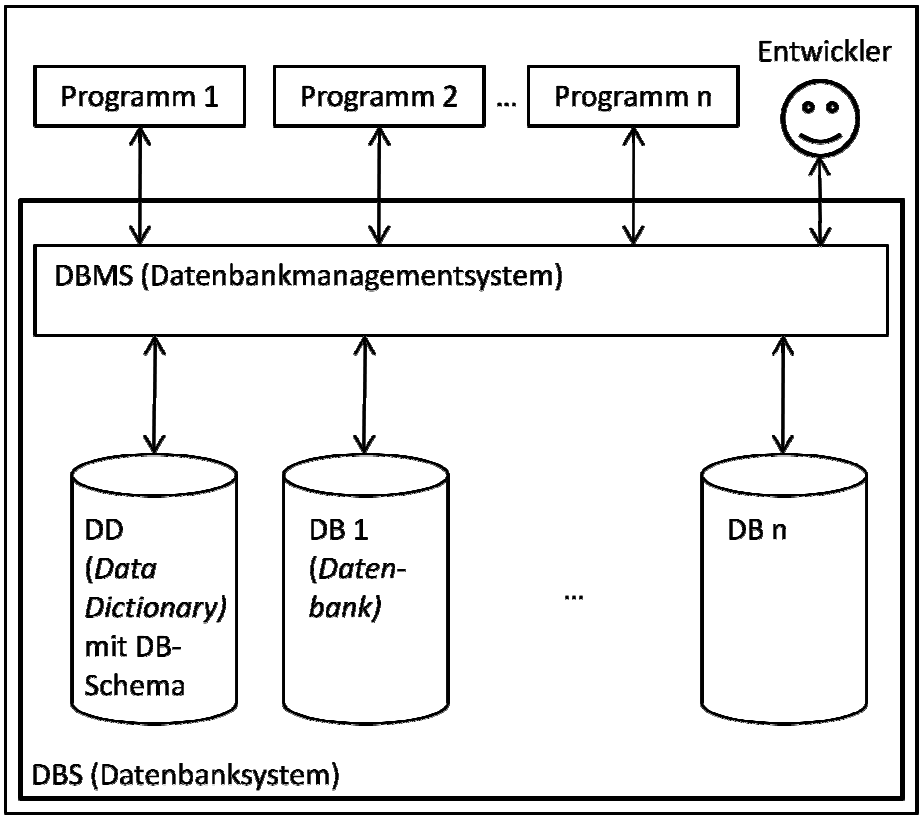

Abbildung 12: Aufbau des Datenbanksystem (vgl. [5], S. 720)

Die Datenbank(en) enthalten dabei die Gesamtheit der Daten für einen Anwendungsbereich. Das Datenbankschema, das den Aufbau der Datenbank(en) beschreibt wird im *Data Dictionary* (DD) gespeichert. Für die zentrale Verwaltung und die Kontrolle, der in der Datenbank oder den Datenbanken abgelegten Datenbestände, ist das Datenbankmanagementsystem (DBMS) zuständig. ([5], S. 721)

Damit das DBS in der Lage ist die zuvor beschriebenen Leistungen zu erbringen und „… damit die Anwendungsprogramme nicht allein für die Semantik der Daten zuständig sind, müssen die Inhalte einer Datenbank durch Angaben über ihre Bedeutung beschrieben werden“ ([5], S. 722). Das *Datenmodell* gibt Aufschluss durch welche Eigenschaften Datenelemente charakterisiert werden können, wie die Struktur der Datenelemente aussehen kann, welche Konsistenzbedingungen einzuhalten sind und welche Operationen zum Speichern, Auffinden, Ändern und Löschen von Datenelementen erlaubt sind (vgl. [5], S. 722).

Wird das Datenmodell auf einen speziellen Einsatzfall angewandt, in diesem Fall die Technologieplanungsanwendung, spricht man von einem Datenbankschema. Es beschreibt den Aufbau einer konkreten Datenbank mit Hilfe einer Definitionssprache.

Für die zu realisierende Web-Anwendung entschied man sich dabei für das relationale Datenmodell. Da es, erstens am weitesten verbreitet ist und somit die größtmögliche Unterstützung erfährt und zweitens war „… das objektorientierte Datenmodell […] bisher nur in Teilbereichen erfolgreich“ ([5], S. 722). In relationalen Datenbanksystemen werden Daten in Form von Tabellen abgespeichert. Jede Tabelle stellt einen sog. Entitätstyp dar, dabei repräsentiert jede Spalte der Tabelle ein Attribut dieses Typs. Jedes Attribut besitzt wiederum einen Attributtyp. In jeder Zeile der Tabelle kann eine Entität des Entitätstyps, welcher als Tupel bezeichnet wird, gespeichert werden ([5], S. 724). Die folgende Abbildung zeigt die Lieferanten-Tabelle des Technologieplanungswerkzeugs in verkürzter Form, wie sie das DBMS phpMyAdmin anzeigt.

Schlüsselattribut

Lieferanten_id	Name	Land	Mail	Strasse
7	Strojimport GmbH (Skoda)	Deutschland	info@strojimport.de	Paul-Friedländer-Strasse
24	Dörries Scharmann Technologie GmbH	Deutschland	roland.wozny@ds-technologie.de	Hugo Junkers Straße

Attribute

Tupel

Abbildung 13: Der Entitätstyp Lieferanten in tabellarischer Form

Einen Ausschnitt des logischen Schemas, d.h. die Definition der Datenstrukturen in Tabellenform zeigt die Abbildung 14.

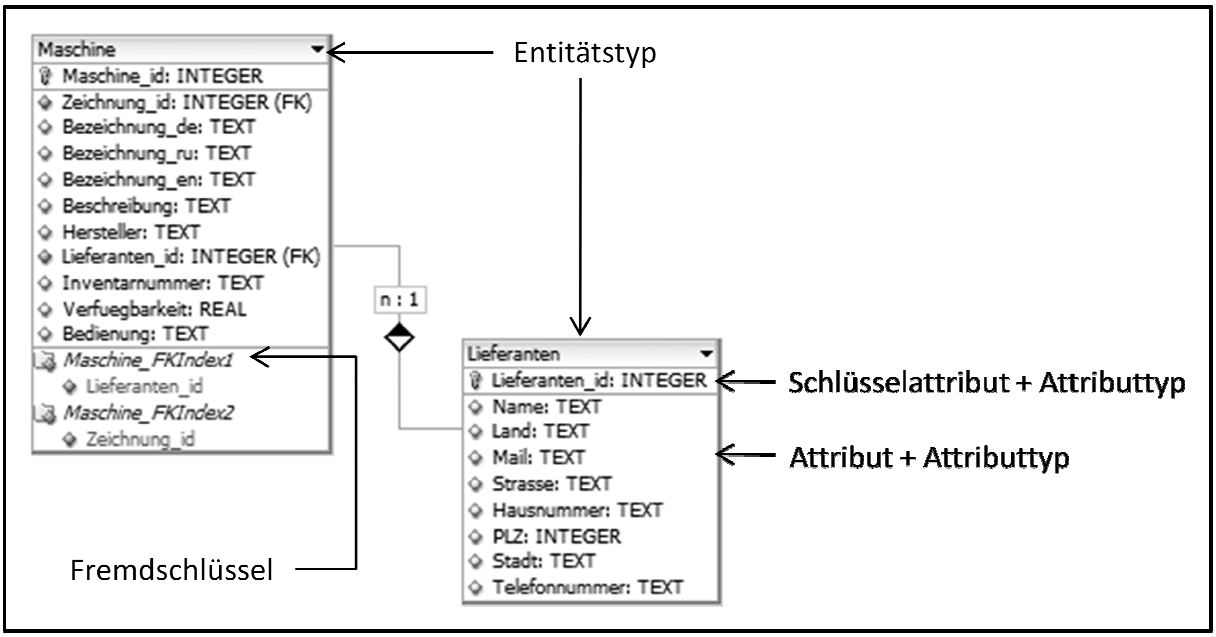

Abbildung 14: Ausschnitt des Logischen Schemas der Technologieplanungsanwendung

Die vollständige Datenstruktur der Technologieplanungsanwendung kann im Anhang A2 eingesehen werden. Jede Entität bzw. Tabelle muss im relationalen Datenmodell durch einen eindeutigen Schlüssel identifizierbar sein. Dieser Schlüssel wird als

Primärschlüssel (siehe Abbildung 14, Schlüsselattribut) bezeichnet und dient zum identifizieren eines Tupels in einer Tabelle. Um Beziehungen zwischen Entitätsmengen (Tabellen) darzustellen, bemächtigt man sich sog. Fremdschlüssel. Abbildung 14 zeigt die 1:n Beziehung zwischen Lieferanten und Maschine. Die Entität Maschine wird dabei um das Schlüsselattribut Lieferanten_id (es kann auch ein Kurzname verwendet werden) der Lieferanten-Tabelle erweitert. Dadurch kann zu jeder Maschine der Lieferant ermittelt werden, wobei ein Lieferant mehrere Maschinen liefern kann. ([5], S. 725)

Die Umsetzung des logischen Schemas in das relationale DBS geschieht mittels Datendefinitionssprache (data definition language, DDL) die vom DBMS zur Verfügung gestellt wird. DDLs sind in der Regel Sprachen der 4.Generation, d.h. sie beschreiben nur das Resultat aber nicht die Aktion die benötigt wird um das Ergebnis zu erreichen (siehe auch [5], S. 719). Als Standard hat sich hier SQL durchgesetzt. Folgendes Beispiel zeigt wie die Tabelle Lieferanten mittels *structured query language* (SQL) erzeugt wird. Es ist Teil des Scripts das aufgerufen wird, wenn ein neues Projekt im Technologie-planungswerkzeug angelegt wird.

```
127 CREATE TABLE IF NOT EXISTS `lieferanten` (
128   `Lieferanten_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
129   `Name` text COLLATE utf8_unicode_ci,
130   `Land` text COLLATE utf8_unicode_ci,
131   `Mail` text COLLATE utf8_unicode_ci,
132   `Strasse` text COLLATE utf8_unicode_ci,
133   `Hausnummer` text COLLATE utf8_unicode_ci,
134   `PLZ` int(10) unsigned DEFAULT NULL,
135   `Stadt` text COLLATE utf8_unicode_ci,
136   `Telefonnummer` text COLLATE utf8_unicode_ci,
137   PRIMARY KEY (`Lieferanten_id`)
138 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

Abbildung 15: Erzeugung einer Lieferanten-Tabelle mittels SQL

Dabei wird zunächst geprüft ob die Tabelle Lieferanten bereits existiert, ist dies nicht der Fall wird sie erzeugt. Nacheinander werden die Attribute Lieferanten_id, Name, Land, Mail etc. mit ihren Attributtypen (z.B. text, int), hinzugefügt. Die Lieferanten_id stellt den Primärschlüssel der Tabelle dar und muss daher immer einen Wert besitzen (wird durch NOT NULL angegeben). Beim Einfügen eines neuen Tupels in die Tabelle erhöht sich der Wert des Primärschlüssels automatisch um den Wert 1 (AUTO_INCREMENT). Außerdem werden alle Attribute vom Typ „text“ in den Zeichensatz „utf8_unicode_ci“ konvertiert.

Nach dieser Vorgehensweise wurde das ganze logische Schema des Technologieplanungswerkzeugs (siehe Anhang A2), in eine reale Datenbank konvertiert. Als Werkzeug kam dabei der DB-Designer 4 von fabFORCE zum Einsatz. Mit diesem wurde die Datenstruktur durch Tabellen definiert und somit ein logisches Schema erstellt. Die Funktionalität des DB-Designers, unter der Angabe von Verbindungsdaten (z.B. DB-Name, Hostname etc.) eine Verbindung zum DBMS herzustellen, ermöglicht das automatische Erzeugen der schematisch vorliegenden DB. Die Software erzeugt dazu selbstständig die nötigen SQL-Befehle und führt diese aus. So konnte innerhalb kurzer Zeit die Datenbank für die Technologieplanungsanwendung erzeugt werden.

4 TECHNISCHE GRUNDLAGEN

4.1 Hypertext Preprocessor (PHP) und eingesetzte Technologien

Die Abkürzung PHP steht für „Hypertext Preprocessor“, dabei handelt es sich um eine weitverbreitete Open Source Skriptsprache, die speziell bei Webentwicklungen zum Einsatz kommt. Ermöglicht wird das durch Fähigkeit PHP in HTML einzubinden, was Web-Entwicklern die Möglichkeit gibt, schnell dynamisch generierte Webseiten zu erzeugen. Von ihrer Syntax her ist sie an C, Java und Perl angelehnt, was sie leicht erlernbar macht. Seit Version 3, die auf ca. 10% aller Webserver im Internet installiert war, wird PHP heutzutage von hundertausenden Entwicklern auf mehreren Millionen Web-Seiten verwendet ([19], S. 3).

Die Entwicklung von PHP begann 1995, damals stand der Begriff noch für Personal Homepage Tools und war vom Ursprung her nichts Anderes als eine Sammlung von Perl-Skripten. Ihr Entwickler Rasmus Lerdorf nutzte Sie eigentlich nur um die Zugriffe auf seinen Web-Auftritt zu protokollieren, schuf aber bald darauf eine umfangreichere Version in der Programmiersprache C. Die Version veröffentliche er unter der Bezeichnung PHP/FI (FI stand für *Form Interpreter*). In Kooperation mit Andi Gutmans und Zeev Suraski, die PHP komplett neu schrieben, entstand 1998 die PHP Version 3 (PHP3). Mit der Entwicklung der *Zend Engine*, die bis heute den Kern von PHP darstellt, wurde die Ausführungsgeschwindigkeit und Sicherheit von PHP entscheidend verbessert. Der Boom des World Wide Web Anfang der 90er Jahre beschleunigte die Verbreitung von PHP enorm und es löste den vorherrschenden Standard Perl bald ab. Mittlerweile ist PHP in der Version 5.3.6 mit der Zend Engine II erschienen und zeichnet sich durch folgende Merkmale aus: ([20])

- PHP ist leicht erlernbar und lässt sich von Neueinsteigern einfach anwenden, bietet daneben aber auch versierten Programmierern einen riesigen Funktionsumfang.
- Da PHP ausschließlich serverseitig ausgeführt wird ist keine Installation auf den Clients erforderlich, das macht es plattformunabhängig. „PHP unterscheidet sich von clientseitigen Sprachen wie Javascript dadurch, dass der Code auf dem Server ausgeführt wird und dort HTML Ausgaben generiert, die an den Client gesendet werden. Der Client erhält also nur das Ergebnis der Skriptausführung, ohne das es möglich ist herauszufinden, wie der eigentliche Code aussieht“ ([21]).

- Dadurch das PHP speziell für den Web-Einsatz entwickelt wurde, lässt es sich problemlos in HTML einbinden. Mittlerweile besteht die Möglichkeit nahezu jede Skriptsprache mit PHP zu kombinieren.
- Neben der Verwendung auf allen gängigen Betriebssystemen (z.B. Linux, Microsoft Windows, Mac OS X, RISC OS), unterstützt PHP auch die Mehrzahl der heute gebräuchlichen Webserver wie Apache, Microsoft Internet Information Server, Personal Web Server, Netscape- und iPLanet-Server, Oreilly Website Pro-Server etc. Außerdem wird eine große Anzahl gängiger Datenbanken unterstützt. Dazu zählen MySQL, Sybase, Unix dbm, Oracle, Solid und viele Weitere. Hinzu kommt das PHP die Kommunikation mit vielen Services, die auf Protokollen wie LDAP, IMAP, SNMP, POP3 oder HTTP basieren, unterstützt. Nähere Informationen liefert [22].

JavaScript

Da wo die Funktionalitäten von PHP oder HTML zum Teil nicht ausreichend waren z.B. bei der Überwachung von Buttons, kam die Skriptsprache JavaScript zum Einsatz. Als eine speziell für dynamische Inhalte in Web-Browsern vorgesehene Sprache, erweitert sie die rudimentären Möglichkeiten der Nutzerinteraktion von HTML. Beispiele für den dynamischen Einsatz von JavaScript sind:

- Dynamische Manipulation des Inhalts und der Struktur von Webseiten z.B. Webseite nach Buttondruck aktualisieren (kein Submit-Button) oder Teile der Webseite Ein- bzw. Auszublenden
- Plausibilitätsprüfung von Formulareingaben
- Animierte Banner oder Laufschriften
- Senden und Empfangen von Daten ohne die Seite neu zu laden

4.2 Entwicklungsumgebung

Für die Entwicklung dieses Software-Projekts kam Eclipse als Entwicklungsplattform zum Einsatz. Dabei handelt es sich um eine benutzerfreundliche, freie (Open-Source) Entwicklungsumgebung, die sich über Plugins und die Integration externer Werkzeuge (z.B. CVS, Apache), optimal an ein Software-Entwicklungsprojekt anpassen lässt. Eclipse unterstützt neben Java, C, HTML auch die Entwicklung von PHP. Für jede dieser Sprachen werden dabei unterschiedliche Versionen bzw. Plugins zur Verfügung gestellt.

Für die Entwicklung von PHP bieten sich die PHP Development Tools (PDT, ehemals IDE) mit folgenden Eigenschaften an:

- Syntax Highlighting
- Echtzeit Code-Überprüfung
- Code completion/Vervollständigung bzw. hinting/Vorschlagen
- Code Formatierung
- Code Outline/Browser (Baumansicht über alle Funktionen und Klassen)
- Integrierte Web Browser Vorschau
- Integration von Apache und MySQL möglich
- Unterstützung für Dokumentation mit Hilfe von phpDocumentor
- Flexibles System für Quellcode Vorlagen (PHP, phpDoc, Smarty, ...)
- Smarty-Plugin (siehe Eclipse Galileo (PDT) mit Smarty-Plugin)
- DBG Debugger Integration möglich ([23])

Für das Projekt Technologieplanungsanwendung war dabei vor allem die Integration von Apache und MySQL von Bedeutung, denn durch diese Erweiterungen ließ sich Eclipse sehr gut an die gestellten Ziele anpassen. Das ist möglich, da Eclipse nur den Kern der Anwendung darstellt der die einzelnen Plugins lädt und deren Funktionalität zur Verfügung stellt. Das installierte WampServer Paket ließ sich somit ohne Probleme mit Eclipse nutzen.

Das WampServer Paket stellt eine Windows Web-Entwicklungsumgebung zur Verfügung. Automatisch werden dabei ein Apache Webserver, PHP und eine MySQL-Datenbank installiert. Damit liefert der WampServer optimale Grundvoraussetzungen für die Anwendungsentwicklung auf PHP Basis.

5 UMSETZUNG

5.1 Navigation

In der Regel bestehen Web-Anwendungen aus mehr als einem Fenster, deshalb ist es notwendig den Zusammenhang zwischen den einzelnen Web-Seiten benutzerfreundlich und aufgabengerecht zu strukturieren. Diese Funktion übernimmt das Navigations-Menü. Es stellt die praktische Realisierung der Inhaltsstruktur dar.

Um die im Abschnitt 3.3 beschriebene Inhaltsstruktur der Technologieplanungs-anwendung zu realisieren, kommt ein Navigations-Menü zum Einsatz (Abbildung 16).

Abbildung 16: Navigations-Menü der Technologieplanungsanwendung(verkürzt dargestellt)

Über dieses Ausklapp-Menü hat der Anwender die Möglichkeit Funktionen aufzurufen oder zur gewünschten Seite zu navigieren. Diese wird dann im Hauptfenster dargestellt. Die Realisierung des Menüs geschieht innerhalb eines HTML-Blockelements. Dieses enthält Listenelemente welche die einzelnen Menüpunkte strukturieren (siehe Abbildung 17). Jedes Navigationselement ist mit einem Link hinterlegt der auf die Hauptseite (main.php) zeigt und die Variablen *content* und *language* übergibt.

```
<div style="font-family: Arial, Helvetica, Sans-Serif;">
  <ul id="nav">
    <li><a href="#"><?php echo $language['Produkte']; ?></a>
        <ul>
            <li><a href="main.php?content=produktliste&language=<?php echo $lang; ?>"><?php echo $language['Anzeigen']; ?></a></li>
            <li><a href="main.php?content=produkt_anlegen&language=<?php echo $lang; ?>"><?php echo $language['Anlegen']; ?></a></li>
            <li><a href="main.php?content=produkt_loeschen"><?php echo $language['Löschen']; ?></a></li>
        </ul>
    </li>
```

Abbildung 17: Ausschnitt aus dem Aufbau des Navigationsmenüs

Je nachdem welchen Wert die Variable *content* enthält, wird die entsprechende Seite dynamisch eingebunden. Für die Entscheidung welche Seite eingebunden wird ist die Hauptseite zuständig. Hier wird zunächst geprüft ob die Variable *content* überhaupt einen Wert enthält. Ist dies der Fall wird die Variable *content* ausgelesen und je nach

ihrem Wert, mittels *include*-Funktion, in die entsprechende Seite eingebunden. Sollte die Variable content keinen Wert enthalten wird eine Fehlermeldung ausgegeben. Abbildung 18 demonstriert beispielhaft die Logik innerhalb der Hauptseite.

```php
<?php
    if(isset($_GET['content'])){ // Pruefen ob der Get Parameter 'content' einen Wert enthält

        switch ($_GET['content'])
            {
            //PRODUKT
            case 'produktliste': include("operation/show/produktliste.php");
            break;
            case 'produkt_anlegen': include("operation/input/produkt_anlegen.php");
            break;
            case 'produkt_aendern_anlegen': include('operation/edit/produkt_aendern_anlegen.php');
            break;
            }
        }else{ // wenn Get Parameter 'content' kein Wert enthaelt
            echo "CONTENT was EMPTY";
        }
    ?>
```

Abbildung 18: Ausschnitt aus main.php: Einbinden der Web-Seiten über die content-Variable

5.2 Design und Ergonomie

Das Design der entwickelten Web-Anwendung entsteht aus einer Kombination des Seiten-Aufbau-Konzepts und der Darstellung von Inhaltsformen (siehe Kapitel 3.3). Die Dreiteilung der Anwendung erfolgt als Blocklayout mittels HTML -Elementen.

Abbildung 19: Web-Design des Technologieplanungswerkzeugs

Für die inhaltliche Gestaltung wurde eine CSS-Vorlage erstellt über die sich das „Aussehen“ der Webseite komfortabel anpassen lässt. In Abbildung 19 wird das Ergebnis gezeigt.

Die Ergonomie des GUI ist ebenfalls ein Punkt, der bei der Gestaltung der Anwendung eine Rolle spielt. Die Realisierung erfolgte unter Beachtung der in EN ISO 9241-10 und EN ISO 14915-1:2000 beschriebenen Grundsätze ([5], S. 554). Beide Normen definieren Qualitätskriterien der Dialoggestaltung:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Diese Kriterien wurden bei der Entwicklung der Benutzeroberfläche festgelegt und umgesetzt. Dabei wurde vor allem Wert auf die Punkte Aufgabenangemessenheit, Steuerbarkeit und Fehlertoleranz gelegt. Die Selbstbeschreibungsfähigkeit und die Erwartungskonformität spielen nur in vereinzelten Fällen eine Rolle. Die beiden letzten Kriterien Individualisierbarkeit und Lernförderlichkeit wurden nicht berücksichtigt bzw. verlieren bei dieser Anwendung ihre Bedeutung.

Die Aufgabenangemessenheit, d.h. die Unterstützung des Nutzers bei der effektiven und effizienten Aufgabenerfüllung, sowie die Steuerbarkeit wurden durch die erfolgreiche Umsetzung der Inhalt- bzw. Navigationsstruktur gewährleistet.

Aufgabe der Fehlertoleranz ist es das, dass beabsichtigte Ergebnis trotz fehlerhafter Eingabe, entweder ohne oder durch minimalen Korrekturaufwand des Nutzers erreicht wird. Das Technologieplanungswerkzeug realisiert das mit Hilfe von Umformungsmechanismen. Rationale Zahlenwerte (z.B. Gewichts- oder Längeneinheiten), die im deutschen Sprachraum durch ein Komma getrennt werden, wandelt die Web-Anwendung automatisch in MySQL konforme Punkte um (siehe Abbildung 20).

```
21
22      $untere_Grenze=$_GET["untere_Grenze"];
23      $untere_Grenze = str_replace(",",".", $untere_Grenze);
24
```

Abbildung 20: Umformungsmechanismus einer rationalen Zahl

Zusätzlich erleichtert diese Umformung die Bedienung für den Nutzer, da dieser gebrochene Zahlenwerte einfach über den Nummernblock der Tastatur eingeben kann.

Ebenfalls zum Aufgabenbereich der Fehlertoleranz gehört das Abfangen von Fehlern. Bei falschen Eingaben werden an der entsprechenden Stelle der Technologieplanungsanwendung Fehlermeldungen erzeugt (siehe Abbildung 21).

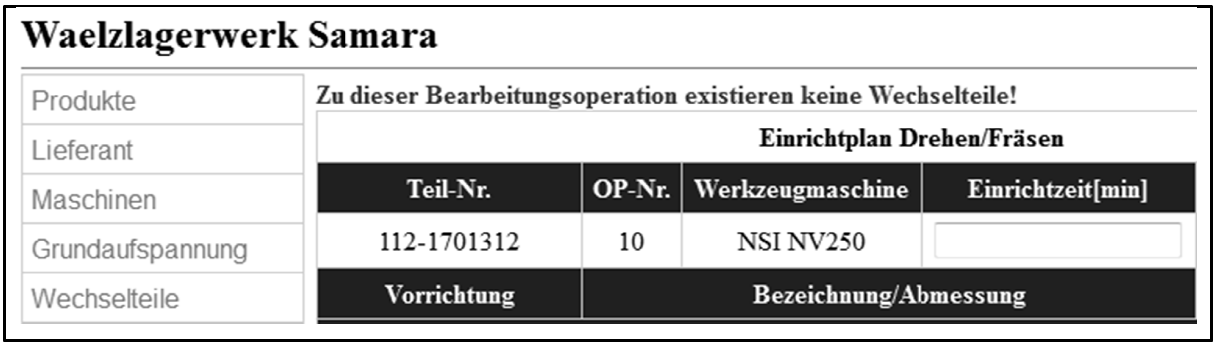

Abbildung 21: Ausgabe und Präsentation von Fehlermeldungen

Um der Anwendung die Fähigkeit der Selbstbeschreibung mitzugeben wurden Schaltflächen, Auswahlmenüs usw. mit sich selbstständig einblendenden Informationsfenstern sog. Tooltips hinterlegt (siehe Abbildung 22).

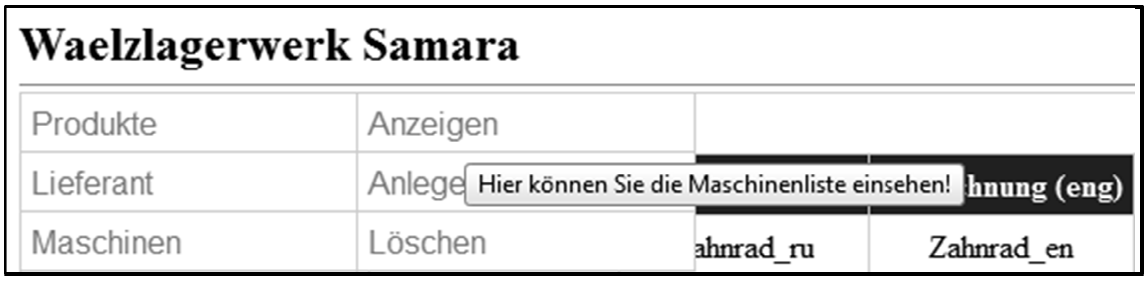

Abbildung 22: Beispiel Tooltip

Den Vorgaben der Erwartungskonformität wurde Rechnung getragen, indem die Darstellung der Elemente der Benutzeroberfläche dem firmeninternen kooperativen Design angepasst wurde. Wie zuvor angesprochen wird die grafische Gestaltung hauptsächlich durch eine CSS-Vorlage bestimmt. Die Abbildungen 23 und 24 veranschaulichen den Einsatz des *Cascading Style Sheet* (CSS).

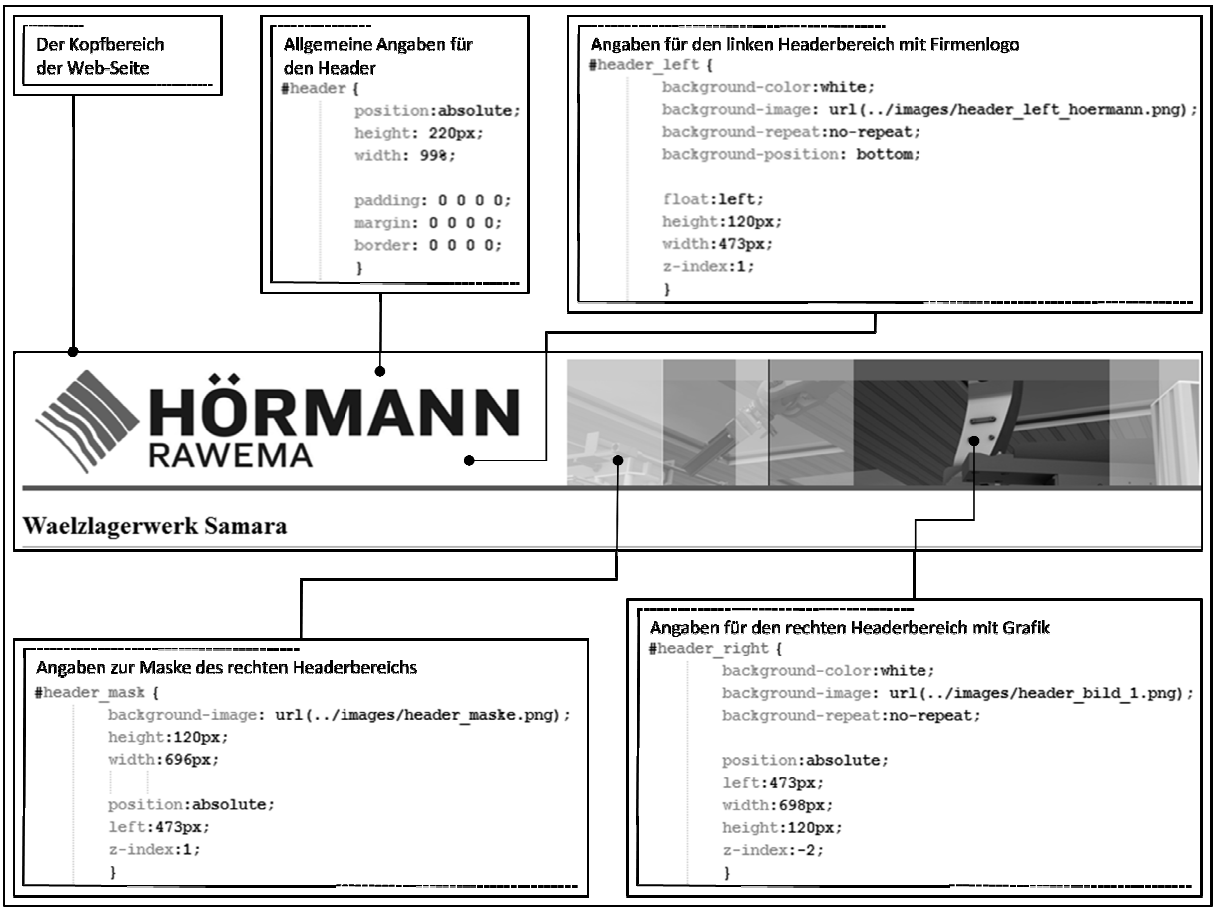

Abbildung 23: Headerbereich der Technologieplanungsanwendung

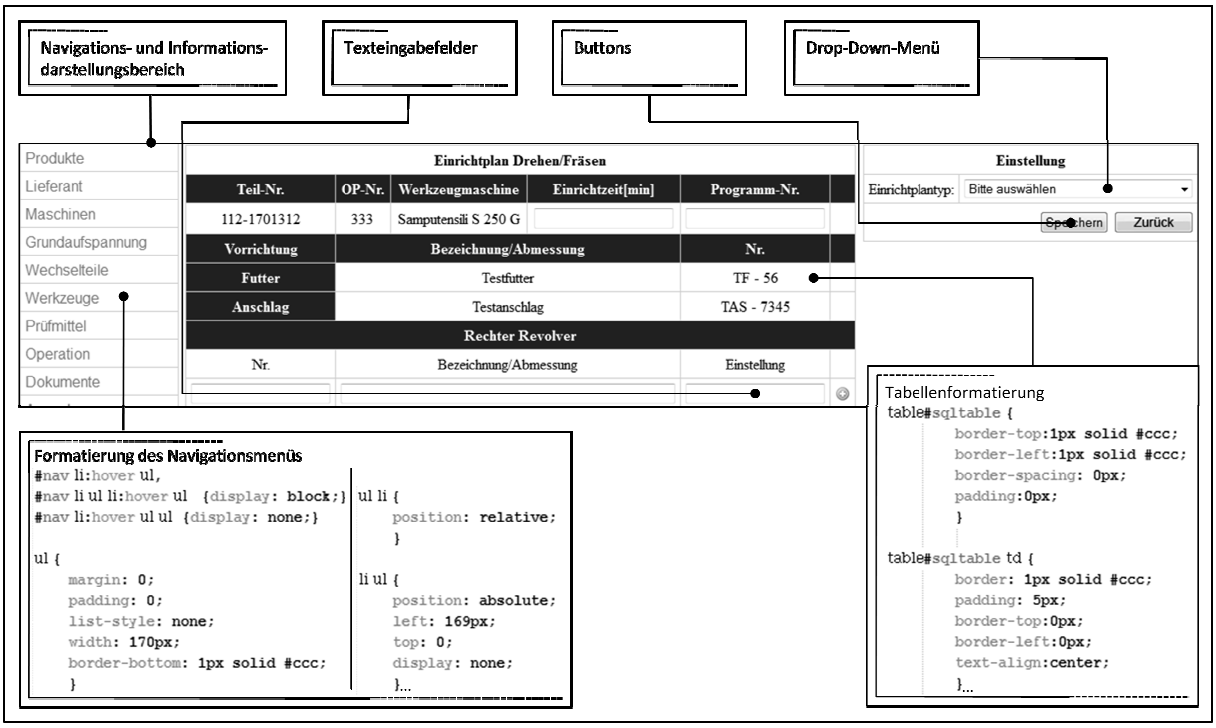

Abbildung 24: Navigations- und Informationsdarstellungsbereich der Technologieplanungsanwendung

5.3 Verarbeitung und Darstellung der Planungsdaten

Neben der Gestaltung der Benutzeroberfläche und des Navigationsmenüs, spielt für die Funktionalität der Technologieplanungsanwendung die korrekte Handhabung der Daten eine wichtige Rolle. Deshalb muss es möglich sein Planungsdaten jederzeit abzurufen, zu aktualisieren oder neue Datensätze hinzuzufügen. Diese Aufgabe übernimmt eine im Hintergrund laufende MySQL-Datenbank. Um auf die Datensätze, in der Datenbank des Technologieplanungswerkzeugs zuzugreifen, d.h. Daten einzutragen, löschen oder zu ändern, wird die Datenmanipulationssprache (DML) SQL genutzt. Eingebettet in ein PHP-Skript werden die SQL-Befehle ausgeführt. Damit der SQL-Befehl ausgeführt werden kann, muss zunächst die Verbindung zur DB hergestellt werden. Die Verbindungsparameter wurden dafür in einer zentralen Konfigurationsdatei abgelegt.

```php
<?php
  //anmeldedaten fuer MySQL
    $userName="root";
    $userPasswort="";
    $serverHost="localhost";
    $datenbankName="samara";
    //Document-Root
    $root="http://localhost/samara/";
    //pfad zum Repository, aufruf nur über Verlinkung

    //data for calculation of production data
    $transfer_time=1200; // transfer time in seconds
    $workingtime=9; // working time per day
    $secure_time=2; // secure for planning
?>
```

Abbildung 25: Zentrale Konfigurationsdatei für die DB-Verbindung

Die Konfigurationsdatei (config.php) wird vor jeder Abfrage eingebunden, dadurch erhält die aufrufende PHP-Seite Zugriff auf deren Inhalt. Zusätzlich kommt die, von den MySQL-Entwicklern zu Verfügung gestellte, MySQLDB-Klasse zum Einsatz. Sie stellt eine Reihe von Funktionen für den Umgang mit der MySQL-DB zur Verfügung. Mit ihrer Hilfe lassen sich Abfragen und Zugriffe teilweise vereinfachen.

```php
    //SQL-Verbindung herstellen
    include("/db/MySQLDB.php");
    include ("/db/config.php");
    $db=new MySQLDB($serverHost,$userName,$userPasswort,$datenbankName);

    //SQL-Abfrage Maschinenfilter
    $maschine_abfrage = "SELECT DISTINCT Maschine.Maschine_id, Maschine.Bezeichnung_de
                         FROM Maschine, Bearbeitungsoperation
                         WHERE Bearbeitungsoperation.Maschine_id=Maschine.Maschine_id";
    $maschine_daten=$db->executeQuery($maschine_abfrage) or die( mysql_error() );
```

Abbildung 26: DB-Abfrage innerhalb einer PHP-Seite

Die abgefragten Datensätze (siehe Abbildung 26) werden in Form eines Arrays gespeichert. Dieses Array lässt sich dann unter Angabe des Zeilenindexes sukzessive auslesen. Meist geschieht das mit Hilfe einer Schleife, die die Array-Daten zeilenweise in Form einer Tabelle ausgibt (siehe Abbildung 27).

```
41  <!-- Tabelleneinträge -->
42  <?php
43  for($i=0; $i<count($maschine_daten);$i++)
44  {
45  echo    "<tr><td>".$maschine_daten[$i][1]."</td>";
46  echo        "<td>".$maschine_daten[$i][2]."</td>";
47  echo        "<td>".$maschine_daten[$i][3]."</td>";
48  echo        "<td>".$maschine_daten[$i][6]."</td>";
49  echo        "<td>".$maschine_daten[$i][7]."</td>";
50  echo        "<td>".$maschine_daten[$i][8]."</td>";
51  echo        "<td>".$maschine_daten[$i][4]."</td>";
52  echo        "<td><a href=".$maschine_daten[$i][9]."/".$maschine_daten[$i][10].">".$maschine_daten[$i][10]."</a></td>";
```

Abbildung 27: Ausgabe der im Array gespeicherten Daten

Auf diese Weise lassen sich die Daten, die in der DB abgelegt sind bearbeiten und darstellen. Die grafische Darstellung der Tabelle erfolgt dabei über das im vorigen Abschnitt angesprochene CSS.

6 ZUSAMMENFASSUNG UND AUSBLICK

6.1 Ergebnisse und Abweichungen

An dieser Stelle soll nochmal ein Blick auf den Entwicklungsverlauf der Web-Anwendung geworfen werden. Am Anfang des Projekts stand die Idee, eine Lösung zu schaffen die die Technologieplanung unterstützt und vereinfacht. Die Entwicklung einer Web-Anwendung welche die geforderten Funktionalitäten, wie die Verwaltung der Stammdaten, unterschiedliche Auswertungsmöglichkeiten zur Verfügung zu stellen oder die Ausgabe von Arbeitsdokumenten, leistet erschien als geeignetes Mittel.

Damit diese Idee umgesetzt werden konnte, war es zunächst notwendig eine genaue Beschreibung des neuen Systems anzufertigen. Das geschah in Form eines Pflichtenhefts. Diese gesammelten Anforderungen bilden die Basis bzw. den Kern der Anwendung. Dabei mussten neben technischen Aspekten, wie die Frage nach der Architektur des Systems, vor allem Entscheidungen zum allgemeinen Vorgehen bei der Entwicklung geklärt werden. Nachdem die Anforderungen definiert waren konnte mit der Konzeption des Technologieplanungswerkzeugs begonnen werden.

Die Entscheidung, das spiralförmige (evolutionäre) Prototyping als Vorgehensmodell zu nutzen, erwies sich als sehr effektiv. Mit jeder neuen Version des Prototyps konnte man genauer spezifizieren, in welche Richtung entwickelt werden muss, gleichzeitig wurde eine Risikobetrachtung vor jeder Phase durchgeführt. Durch das Einbeziehen der Nutzer des Systems in den Entwicklungsprozess, konnte die Anwendung schon zeitig an deren Bedürfnisse angepasst werden.

Da der Einfluss zukünftiger Nutzer des Systems auf den Entwicklungsprozess sehr hoch war, gab es letztendlich viele Abweichungen von den ursprünglich definierten Anforderungen. Das führte dazu, dass man manchmal den Ideenfluss der Nutzer bremsen oder deren Wünsche priorisieren musste. In seltenen Fällen war auch die Umsetzung aus technologischer Sicht nicht möglich oder führte zu Konflikten mit anderen Anforderungen.

6.2 Erweiterungs- und Verbesserungsvorschläge

Trotz dessen, dass das System nach Abschluss dieser Arbeit seine komplette Funktionalität besitzt, bestehen aus der Sicht des Autors und der Nutzer noch Erweiterungsvorschläge.

Die Erweiterungsvorschläge betreffen in erster Linie die Dokumentenerstellung und die Exportfunktionen. Es soll zukünftig möglich sein sämtliche Auswertungen aus der Anwendung heraus in eine Excel-Tabelle zu exportieren. Einricht- und Arbeitspläne die bisher nur einzeln ausdruckbar sind, sollen um eine PDF-Exportfunktion erweitert werden. Außerdem soll die Möglichkeit bestehen, dass alle Dokumente eines Prozesses automatisch ausgegeben werden.

Eine Verbesserungsmöglichkeit besteht in der Optimierung der ergonomischen Gestaltung des Systems. Dies betrifft insbesondere die Darstellung der Informationen in Form von Tabellen. Mit der wachsenden Anzahl, bestehender Datensätze entstehen zum Teil sehr lange Tabellen. Durch Filter Optionen wurde bereits versucht die Ansicht zu optimieren, allerdings besteht hier noch Ausbaubedarf um die Ansicht in einem benutzerfreundlichen Rahmen zu halten.

Eine weitere Verbesserungsmöglichkeit besteht in der Erweiterung der Datensätze um zusätzliche Informationen. Damit würde sich die Auswertung verbessern lassen und es bestände die Möglichkeit zusätzliche Informationen zu gewinnen.

Der praktische Einsatz des Technologieplanungswerkzeugs wird zeigen welche Erweiterungen und Verbesserungen noch implementiert werden müssen, um die Software zu einem vollwertigen Produkt zu machen. Aus Sicht des Autors sollte aber nicht immer weiter an dem System gearbeitet werden, sondern man muss in absehbarer Zeit eine Entscheidung treffen, das Projekt erfolgreich abzuschließen. Sonst besteht die Gefahr das aus dem Projekt ein ewiges Projekt wird.

LITERATURVERZEICHNIS

[1] Kappel, G.; Pröll, B.; Reich, S.; Retschitzegger, W.: Web Engineering – Systematische Entwicklung von Web-Anwendungen, 2004, Heidelberg: dpunkt.verlag.

[2] Berners-Lee, T.: WWW: Past, Present and Future, IEEE Computer, 1996.

[3] Hitz, M.; Kappel, G.; Reschitzegger, W.; Schwinger, W.: Ein UML-basiertes Framework zur Modellierung ubiquitärer Web-Anwendungen, 2002, Wirtschaftsinformatik, 44 (3).

[4] Cutter Consortium; Poor Project Management Number-one Problem of Outsourced E-projects, Cutter Research Briefs, 2000, www.cutter.com/research/2000/crb001107.html, [letzter Besuch: 2011-06-15].

[5] Balzert, H.: Lehrbuch der Software-Technik: Software-Entwicklung, 2. Auflage, 2000, Heidelberg; Berlin: Spektrum, Akad. Verl.

[6] Kruchten, P.: The 4+1 View Model of Architecture, 1995, IEEE Software.

[7] Wallace, D.;Ragget, I.; Aufgang, J.: Extreme Programming for Web Projects, 2002, Addison-Wesley.

[8] Fritzsche, M; Keil, P.: Kategorisierung etablierter Vorgehensmodelle und ihre Verbreitung in der deutschen Software-Industrie, 2007, München: Technische Universität München.

[9] Powell, T.; Jones, D.; Cutts, D.: Web Site Engineering: Beyond Web Page Design, 1998, Prentice Hall.

[10] Brooks, F. P.:No Silver Bullet: Essence and Accidents of Software Engineering, 1987, IEEE Computer.

[11] Royce, W.: Managing the Development of Large Software Systems: Concepts, 1970, WESCON Proceedings.

[12] Pomberger, G.; Blaschek, G.: Software – Engineering: Prototyping und objektorientierte Software - Entwicklung, 2., überarb. Aufl. – München, 1996, Wien: Hanser.

[13] Sommerville, I.: Software Engineering, 8., aktualisierte Auflage, 2007, München: Pearson Studium.

[14] 830-1998 –IEEE Standart: Recommended Practice for Software Requirements Specifications:, 1998, IEEE Standards Association, http://standards.ieee.org/findstds/standard/830-1998.html, [letzter Besuch 2011-06-24].

[15] Boehm, B. W.: A Spiral Model of Software Development an Enhancement, 1988, IEEE Computer.

[16] Starke, G.: Effektive Software Architekturen – Ein praktischer Leitfaden, 2002, Wien: Hanser.

[17] Jacobsen, I.; Booch, G.; Rumbaugh, J.:The Unified Software Development Process, 1999, München: Addison-Wesley.

[18] Anastopoulos, M.; Romberg, T.: Referenzarchitekturen für Web-Applikationen, Projekt Application2Web, Forschungszentrum Informatik (FZI), 2001, http://app2web.fzi.de/, [letzter Besuch 2011-06-28].

[19] The PHP Group: PHP-Handbuch, 2011, http://de3.php.net/manual/de/preface.php, [letzter Besuch 2011-06-29].

[20] The PHP Group: Die Geschichte von PHP, 2011, http://de3.php.net/manual/de/history.php.php, [letzer Besuch 2011-07-08].

[21] The PHP Group: PHP-Handbuch, 2011, http://de3.php.net/manual/de/intro-whatis.php, [letzter Besuch 2011-07-09].

[22] The PHP Group: PHP-Handbuch, 2011, http://de3.php.net/manual/de/intro-whatcando.php, [letzter Besuch 2011-07-09].

[23] php::bar: PHP Development Tools Framework for the Eclipse Platform, 2009, http://www.phpbar.de/w/Eclipse_PHP_IDE_Project, [letzter Besuch 2011-06-22].

GLOSSAR

CGI (Common Gateway Interface)

Das **Common Gateway Interface** (**CGI**) ist ein Standard für den Datenaustausch zwischen einem Webserver und dritter Software, die Anfragen bearbeitet. CGI ist eine schon länger bestehende Variante, Webseiten dynamisch bzw. interaktiv zu machen, deren erste Überlegungen auf das Jahr 1993 zurückgehen.

CORBA (Common Object Request Broker Architecture)

Die Common Object Request Broker Architecture (CORBA, engl. *Allgemeine Architektur für Vermittler von Objekt-Anforderungen*) ist eine Spezifikation für eine objektorientierte Middleware, deren Kern ein sog. Object Request Broker, der ORB bildet, und die plattformübergreifende Protokolle und Dienste definiert. Sie wird von der Object Management Group (OMG) entwickelt. CORBA-konforme Implementierungen vereinfachen das Erstellen verteilter Anwendungen in heterogenen Umgebungen.

CSS (Cascading Style Sheets)

Nach den vom W3C (World Wide Web Consortium) festgelegten Webstandards, sind Seiteninhalte und deren Formatierung strikt zu trennen. Das HTML-Dokument enthält die Inhalte, während die Stylesheets die Gestaltung übernehmen. Im Idealfall werden die CSS-Angaben in eine externe Datei geschrieben, also ausgelagert.

CVS (Concurrent Version System)

CVS steht für „Concurrent Version System“ und ist ein System zur Versionsverwaltung, entwickelt für Softwareprojekte. Mit dem CVS können mehrere Nutzer gleichzeitig und online an einem Projekt arbeiten, auch gleichzeitig an einer Datei. Die Aufgabe des CVS ist es, die Änderungen im Quelltext (inkl. Bugs) nachvollziehbar zu machen, zu dokumentieren.

GUI (Graphical User Interface)

Eine grafische Benutzeroberfläche ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt.

Legacy System (dt. Altsystem)

Innerhalb der Anwendungslandschaft eines Unternehmens sind es zumeist großrechnerbasierte Individualentwicklungen, die sich oft durch unzureichende Dokumentation, veraltete Betriebs- und Entwicklungsumgebungen, zahlreiche Schnittstellen und hohe Komplexität auszeichnen. Die dort anzutreffende zentrale Daten- und Funktionshaltung galt seit der Client/Server-Euphorie als überholt.

ANLAGEN

A1 PFLICHTENHEFT

Benutzeroberfläche

/B10/ Standardmäßig sind die Richtlinien für firmeninterne Anwendungen zu beachten.

/B20/ Die Bedienung muss für mehrere auf dem Markt gängige Web-Browser realisiert werden.

/B30/ Die Web-Seite wird in drei Frames bzw. einen oberen, einen unteren Frame und das Navigationsmenü unterteilt.

/B30/ Die Bedienungsoberflächen sind auf Maus- und Tastaturbedienung auszulegen

/B40/ „ISO 9241-110: 2006 Ergonomische Anforderungen für Bürotätigkeiten und Bildschirmgeräte“ ist zu beachten.

Nicht Funktionale Anforderungen

Das System soll als Web-Applikation (Client-Server Applikationen) realisiert werden, die nur für den internen Gebrauch verwendet wird.

- es soll eine plattformunabhängige Web-Applikation entwickelt werden
- es soll ein Projektverwaltungsmechanismus eingebunden werden

Technische Produktumgebung

Das Produkt ist Client/Server-fähig.

Software:

- Server-Betriebssystem: Das Server-Betriebssystem ist plattformunabhängig. Es muss aber einen funktionsfähigen Web-Server zur Verfügung stellen, der PHP fähig ist, z.B. Wamp oder Xampp -Server.
- Client-Betriebssystem: Weil das System plattformunabhängig realisiert werden soll, ist die einzige Client-Voraussetzung, dass der Client einen funktionsfähigen Browser (Firefox, Internet Explorer) auf seiner Maschine installiert hat.

Hardware:

- Server: PC
- Client: Gerät mit Grafikbildschirm und installiertem Browser mit aktiviertem JavaScript

Orgware:

- Netzverbindung des Client-Rechners zum Server-PC

Spezielle Anforderungen an die Entwicklungsumgebung

Die Anforderungen an die Entwicklungsumgebung sollen nicht von der Produktumgebung abweichen. Auf der Entwicklungsmaschine müssen daher die gleichen Voraussetzungen bei der Entwicklung geschaffen werden, wie sie später auf Client-Seite vorherrschen d.h. gleiche Browser/-versionen und Einstellungen.

Gliederung der Teilprodukte

Es werden von Anfang an mehrere Teilprodukte geplant. Die ersten prototypischen Entwicklungen sollen den Weg zur Entwicklung der ersten Version des Systems zeigen. Die später folgenden Prototypen sollen die weitere Entwicklung in kleinen Schritten ermöglichen.

Nach Fertigstellung der Grundfunktionalität ist es möglich das zusätzliche Funktionen hinzugefügt werden. Dies sollte jedoch erst nach der Version 2 geschehen.

Funktion	...	System v. 1.0	...	System v. 2.0	...
/F10/ Projekt erstellen / löschen	→	X	→	X	
/F20/ Informationen eintragen / löschen	→	X	→	X	
/F30/ Informationen abrufen	→	X	→	X	
/F40/ Informationen ändern	→		→	X	
/F50/ Auswertung	→		→	X	
/F60/ Filter	→		→	X	
/F70/ Export	→		→		→

Tabelle A1: Gliederung der Teilprodukte

Produktdaten

Die Produktdaten beschreiben welche Daten vom oder im System gespeichert werden. Dabei werden die Daten in 9 Datengruppen unterteilt: Produktdaten, Lieferantendaten, Maschinendaten, Aufspannungsdaten, Wechselteildaten, Werkzeugdaten, Prüfmitteldaten Prüfoperationsdaten und Bearbeitungsoperationsdaten.

Produktdaten /D10/
Zu den Produktdaten gehören alle Informationen über das Produkt: Bezeichnung, Nummer, Beschreibung, Jahresbedarf, Fertigteilgewicht usw.

Lieferantdaten /D20/
Zu den Lieferantendaten gehören alle Informationen über den Lieferanten: Name, Land, Adresse, Telefonnummer usw.

Maschinendaten /D30/
Zu den Maschinendaten gehören alle Informationen über die Maschine: Bezeichnung, Beschreibung, Hersteller, Lieferant usw.

Aufspannungsdaten /D40/
Zu den Aufspannungsdaten gehören alle Informationen über die Aufspannung: Bezeichnung, Nummer, Wechselzeit, Maschine usw.

Wechselteildaten /D50/
Zu den Wechselteildaten gehören alle Informationen über die Wechselteile: Aufspannung, Aufbau, Bezeichnung, Nummer, Wechselzeit usw.

Werkzeugdaten /D60/
Zu den Werkzeugdaten gehören alle Informationen über die Werkzeuge: Bezeichnung, Beschreibung, Standzeit, Revolverposition usw.

Prüfmitteldaten /D70/
Zu den Prüfmitteldaten gehören alle Informationen über die Prüfmittel: Bezeichnung, Typ, Serialnummer, Lieferant usw.

Bearbeitungsoperationsdaten /D80/
Zu den Bearbeitungsoperationsdaten gehören alle Informationen über die Bearbeitungs-operationen: OP-Nr., Bezeichnung, Produkt, Maschine usw.

Prüfoperationsdaten /D90/
Zu den Prüfoperationsdaten gehören alle Informationen über die Prüfoperationen: Prüfmittel, Produkt, Bearbeitungsoperation, Nummer, Maß usw.

Produktleistungen

Anmeldung:

Die Dauer des Logins in ein Projekt sollte im Bereich von 1-2 Sekunden liegen.

Akkumulation:

Bei fehlerhaften Eingaben wird dem User eine entsprechende Fehlermeldung angezeigt.

Toleranz:

Bei fehlerhaften Eingaben soll die Korrektur der Eingabedaten möglich sein, ohne die Eingabe wiederholt vornehmen zu müssen.

Qualitätsanforderungen

Produktqualität	Sehr Gut	Gut	Normal	Nicht Relevant
Funktionalität				
Angemessenheit		x		
Richtigkeit		x		
Interoperabilität		x		
Ordnungsmäßigkeit		x		
Sicherheit				
Zuverlässigkeit				
Reife			x	
Fehlertoleranz		x		
Wiederherstellbarkeit			x	
Benutzbarkeit				
Verständlichkeit		x		
Erlernbarkeit		x		
Bedienbarkeit		x		
Effizienz				
Zeitverhalten			x	
Verbrauchsverhalten			x	
Änderbarkeit				
Analysierbarkeit			x	
Modifizierbarkeit			x	
Stabilität		x		
Prüfbarkeit		x		

Tabelle A2: Übersicht Qualitätskriterien

Produktfunktionen

	/F10/ Projekt erstellen / löschen	/F20/ Informationen eintragen / löschen
Prozess	Projekt erstellen / löschen	Informationen eintragen / löschen
Ziel	Projekt wird erstellt / gelöscht	Datensatz wird erstellt / gelöscht
Kategorie	Primär	Primär
Vorbedingung	-	User muss in Projekt sein
Nachbedingung Erfolg	Projekt wurde erstellt / gelöscht	Datensatz wurde eingetragen / gelöscht
Nachbedingung Fehlschlag	Fehlermeldung	Fehlermeldung
Auslösendes Ereignis	Projektname u. DB-Name eintragen	Eintragen der Daten / drücken des Löschen-Buttons
Beschreibung	Projektname und DB-Name prüfen	Datensatz ID zuweisen und Daten eintragen / Datensatz ID lesen und entsprechenden Datensatz in DB löschen
Erweiterung	-	-

Tabelle A3: Übersicht Produktfunktionen 1

	/F30/ Informationen abrufen	/F40/ Informationen ändern
Prozess	Informationen darstellen	Informationen ändern
Ziel	Informationen werden dargestellt	Datensatz wird geändert
Kategorie	Primär	Primär
Vorbedingung	User muss in Projekt sein	User muss in Projekt sein
Nachbedingung Erfolg	Informationen werden dargestellt	Datensatz wurde geändert
Nachbedingung Fehlschlag	Fehlermeldung	Fehlermeldung
Auslösendes Ereignis	Auswahl einer Anzeige-Funktion	Eintragen der Daten / drücken des Löschen-Buttons
Beschreibung	Kategorie prüfen / Filter prüfen	Datensatz ID prüfen und Daten eintragen
Erweiterung	Anzeigeergebnis kann durch Filter angepasst werden	-

Tabelle A4: Übersicht Produktfunktionen 2

	/F50/ Auswertung	/F60/ Filter
Prozess	Auswertung darstellen	Filter anwenden
Ziel	Auswertung wird dargestellt	Anzeigeinformationen werden gefiltert
Kategorie	Primär	Primär
Vorbedingung	User muss in Projekt sein	User muss in Projekt sein / Informationen müssen bereits abgerufen sein
Nachbedingung Erfolg	Auswertung wird dargestellt	Anzeige des gefilterten Datensatzes
Nachbedingung Fehlschlag	Fehlermeldung	Fehlermeldung
Auslösendes Ereignis	Auswahl einer Auswertungs-Funktion	Filter auswählen
Beschreibung	Kategorie prüfen	Filteroptionen prüfen
Erweiterung	-	-Filter hinzufügen

Tabelle A5: Übersicht Produktfunktionen 3

	/F70/ Export
Prozess	Informationen exportieren
Ziel	Informationen werden exportiert
Kategorie	Primär
Vorbedingung	User muss in Projekt sein
Nachbedingung Erfolg	Informationen werden exportiert
Nachbedingung Fehlschlag	Fehlermeldung
Auslösendes Ereignis	Auswahl einer Export- oder Druck-Funktion
Beschreibung	Kategorie prüfen / Exportart prüfen
Erweiterung	-

Tabelle A6: Übersicht Produktfunktionen 4

Zielbestimmung

Es soll ein Technologieplanungswerkzeug entwickelt werden. Dabei soll es möglich sein alle relevanten Daten anschaulich darzustellen und bei Bedarf zu ändern. Außerdem soll das Werkzeug Möglichkeiten der Auswertung und Analyse der Daten bereitstellen. Ein Exportfunktion wäre wünschenswert.

Musskriterium

Hier werden die Systemleistungen definiert, die auf jeden Fall erfüllt werden müssen:

- Projekt-Verwaltung

- Verwaltung der Stammdaten

Sollkriterium

Die Erfüllung dieser Leistungen soll angestrebt werden:

- Einzel-Export mittels Druck-CSS
- Auswertung
- Sortierung

Wunschkriterium

Wunschkriterien sind nicht unbedingt zu erfüllen, ihre Realisierung ist wünschenswert:

- Export von Auswertung- und Dokumenten nach Excel-Tabellen
- Export von Auswertung- und Dokumenten als PDF-Dokument

Abgrenzkriterium

Abgrenzkriterien setzen die Grenzen des zu entwickelnden Systems:

- Einsatz im Backoffice-Bereich (Firmenintern)

Produkteinsatz

Produkt dient als Werkzeug zur Technologieplanung

Anwendungsbereiche

Planungsbereich

A2 DATENBANKSTRUKTUR

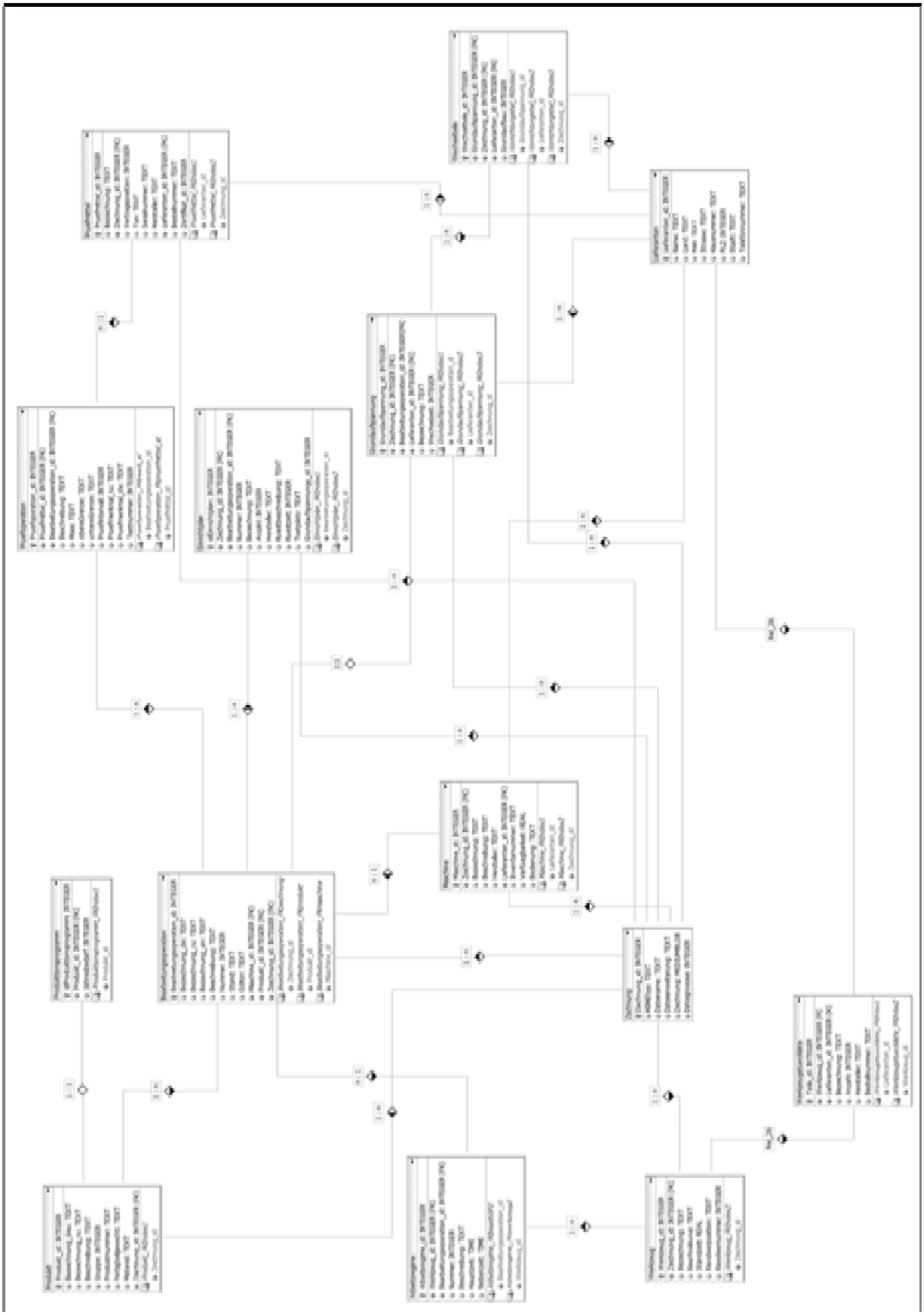

Abbildung A1: Datenbankstruktur der Technologieplanungsanwendung